.UBAfiuba
FACULTAD DE INGENIERÍA

Transformer Decoders

Docentes:
Esp. Ing Abraham Rodriguez - FIUBA
Esp. Ing Ezequiel Guinsburg - FIUBA

Programa de la materia

1. Repaso de Transformers, Arquitectura y Tokenizers.
2. Arquitecturas de LLMs, Transformer Decoder.
3. Ecosistema actual, APIs, costos, HuggingFace y OpenAI.
4. MoEs, técnicas de prompts, evaluación de LLMs.
5. Modelos locales y uso de APIs.
6. RAG, vector DBs, chatbots y práctica.
7. Agentes, fine-tuning y práctica.
8. Generación multimodal.

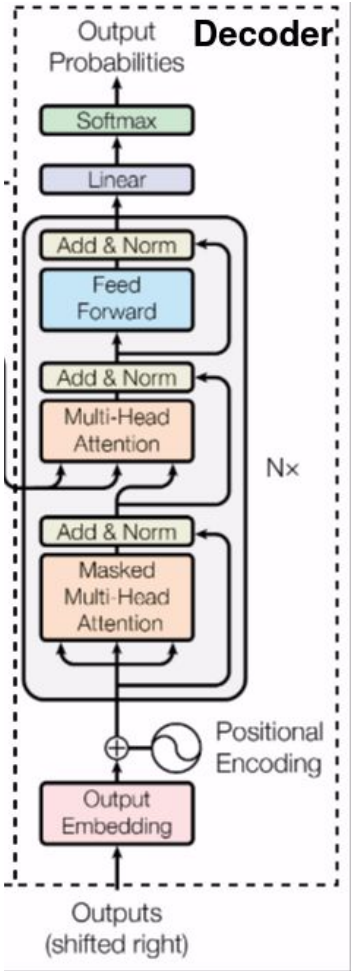
Transformers Decoders

Las LLMs y modelos generativos basados en Transformers son normalmente Transformer Decoders. Las LLMs son realmente una arquitectura de Decoder llamada **Generative Pretrained Transfomer (GPT).**

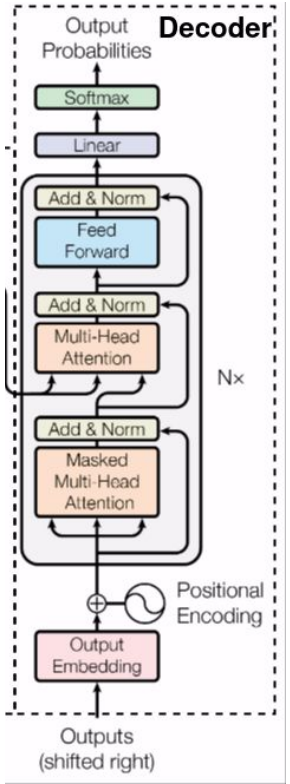
El Decoder se encarga de decodificar un espacio latente, en el caso de la arquitectura completa, es el espacio generado por el encoder.

El espacio latente es un espacio de información que consta de las características de los datos como la relación entre palabras.

El espacio latente es utilizado por el decoder para “generar” información a vectores de entrada, por ejemplo completar secuencias de texto.



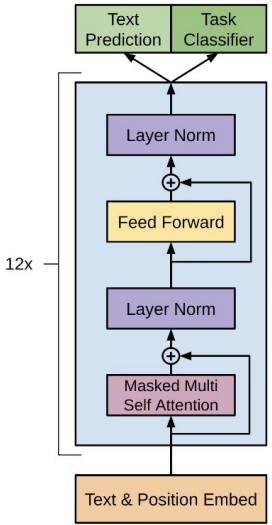
Generative Pretrained Transformer (GPT)

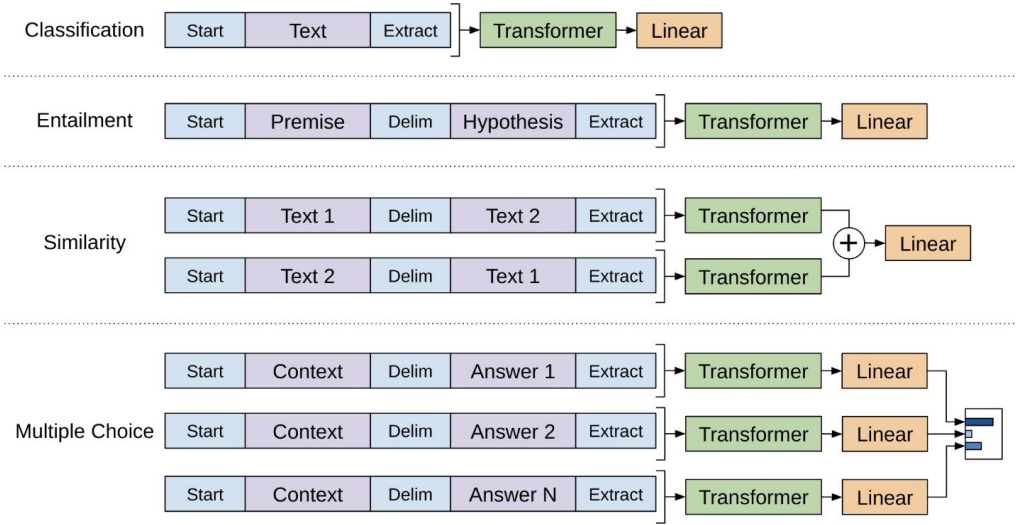


GPT es la arquitectura que trajo como consecuencia modelos del estado del arte como chatGPT (closed-source), Llama, Mistral, Gemma (open-source). Así como modelos multimodales como CLIP.

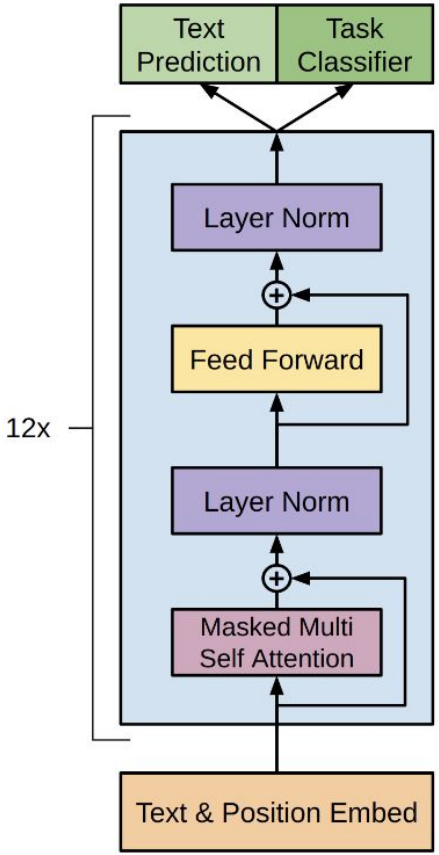
En 2018 OpenAI presentó el paper Improving Language Understanding by Generative Pre-training. Donde se introdujo el concepto de Generative Pretraining, utilizando la arquitectura del **Transformer Decoder ligeramente modificado**.

El paper demuestra distintos objetivos de entrenamiento de GPT.





GPT Unsupervised Pre-training



Dado un corpus de tokens $U = \{u_1, \ldots, u_n\}$, se utiliza un objetivo de modelado estándar de lenguaje (causal) para maximizar la verosimilitud:

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

donde *k* es el tamaño de la ventana de contexto, la probabilidad condicional P es modelada usando una red neuronal con parámetros Θ entrenables mediante SGD.

La red neuronal referida es el **Transformer Decoder**, el cual da como resultado una distribución de probabilidad de tokens de salida.

$$h_0 = UW_e + W_p$$
$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$
$$P(u) = \texttt{softmax}(h_n W_e^T)$$

$U = (u_{-k}, \ldots, u_{-1})$ vector de contexto de tokens

W_e matriz de token embeddings

W_p position embeddings

n # de capas

GPT Unsupervised Pre-training

PRE ENTRENAMIENTO	
Característica	**Detalles**
Arquitectura	12 capas de transformer decoder.
Dimensiones	768 embeddings 12 attention heads 3072 MLP.
Entrenamiento	Optimizador Adam LR Max: 2.5e-4 100 epochs
BatchSize	64 minibatch de 512 tokens.
Dropout	0.1 (embeddings)
Regularización	L2 (weight 0.01) Dropout (p = 0.1)
Funcion de Activacion	GELU
Tokenizer	BPE con 40k tokens (merges).
Corpus	BooksCorpus

El objetivo del preentrenamiento es simplemente predecir la siguiente palabra de una secuencia, con la meta de aprender y estructurar patrones de lenguaje natural.

Next Token Prediction

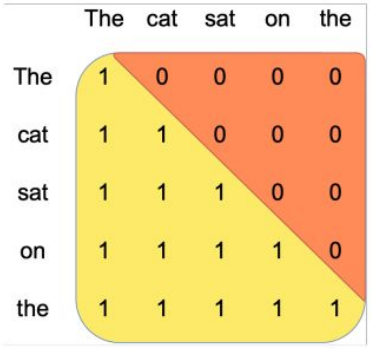
$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

Causal Attention

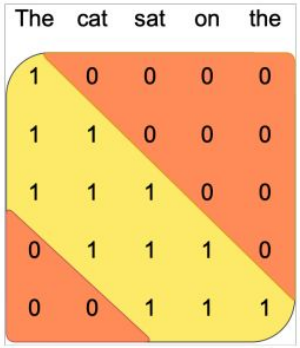
Debido a que el modelado causal implica la generación de un token dado el anterior, en el transformer no es nada más que el enmascaramiento de la matriz de atención, siendo en realidad el ya conocido masked multi-headed attention.

Understanding and coding self-attention

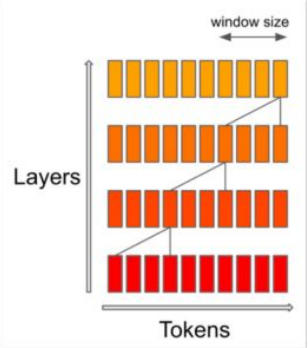
$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$



Vanilla Attention



Sliding Window Attention



Effective Context Length

GPT Supervised Fine-Tuning (SFT)

Luego de entrenar al modelo con el objetivo de maximizar la verosimilitud, se asume que se cuenta con un dataset de secuencias de tokens $X = \{x_1, \ldots, x_m\}$ etiquetado con y. Las entradas son pasadas por un modelo preentrenado para obtener la última activación h_l^m del transformer block. Se agrega una capa lineal con parámetros W_y para predecir a y.

$$P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y)$$

Esto genera un nuevo objetivo, siendo maximizar la verosimilitud:

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m)$$

En otras palabras consta de maximizar la probabilidad de la etiqueta con respecto a las secuencias.

Adicionalmente para acelerar y mejorar la generalización y convergencia, se define el lagrangiano que es maximizar la verosimilitud sobre el dataset dado un peso λ.

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

GPT Supervised Fine-Tuning (SFT)

Understanding and Using Supervised Fine-Tuning (SFT) for Language Models

FINE TUNING	
Característica	**Detalles**
Entrenamiento	Optimizador Adam LR Max: 6.25e-5 3 epochs
BatchSize	32 minibatch de 512 tokens.
Dropout	0.1 (embeddings)
Regularización	Dropout (p = 0.1)

Generative Pretrained Transformer (GPT-2)

GPT-2 es la arquitectura que definió a **Chat LLMs** y a los modelos generativos actuales.

En 2019 OpenAI presento el paper Language Models are Unsupervised Multitask Learners. Donde se reutilizo la arquitectura de GPT muy ligeramente modificada. Básicamente es mas grande y robusta, el cambio más significativo es un nuevo dataset diverso en contenido.

Gracias a su capacidad aumentada es capaz de hacer Zero-shot lo que significa que es capaz de hacer tareas las cuales no fue entrenado, dando la cualidad de generalizar tareas.

Característica	GPT-1	GPT-2
Parámetros	110 milliones	hasta 1.5 billones
Hidden Dimensions	768	1024 (modelos grandes)
Attention Heads	12	16
Training Dataset	BooksCorpus (5GB)	WebText (45GB)
Text Length	Max 512 tokens	Max 1024 tokens
Zero-shot Learning	No	Si
Text Generation	Decente y simple	Coherente y largo

GPT-2 Marcó la necesidad de datasets de alta calidad, más grandes, así como modelos más robustos +7B, 21B, 90B...

The Illustrated GPT-2

Guia explicativa

Recomendable leer!

Generative Pretrained Transformer (GPT-3)

En 2020 OpenAI presentó el paper Language Models are Few-Shot Learners. Donde se reutilizo la arquitectura de GPT-2 modificada. Utiliza un nuevo mecanismo de atención optimizado similar a Sparse Transformer. Entrenaron 8 versiones del modelo.

GPT-3 “davinci” trato de comprobar la hipótesis de que escalar un modelo mejora el rendimiento de múltiples tareas sin necesidad de realizar fine-tuning, buscando que el modelo sea capaz de generalizar solamente entrenando con un dataset diverso

El entrenamiento de GPT-3 fue computacional y energéticamente costoso, consumió miles de petaflops/día durante preentrenamiento.

GPT-3 no utilizo **fine-tuning**, en su lugar utilizó **Few-shot, One-shot, Zero-shot.**

Resumen del paper

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or “GPT-3”	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

Dataset	Quantity (tokens)	Weight in training mix
Common Crawl (filtered)	410 billion	60%
WebText2	19 billion	22%
Books1	12 billion	8%
Books2	55 billion	8%
Wikipedia	3 billion	3%

Generative Pretrained Transformer (GPT-3)

Segun LambdaLabs GPT-3 tuvo un costo aproximado de $4.6M.

Recomendable leer!

GPT-3 Zero-shot y One-shot

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
1  Translate English to French:      <-- task description
2  cheese =>                         <-- prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
1  Translate English to French:      <-- task description
2  sea otter => loutre de mer        <-- example
3  cheese =>                         <-- prompt
```

Zero-shot implica que el modelo es capaz de realizar tareas sin demostraciones previas, gracias a las grandes cantidades de datos.

Prompt Engineering Guide: Zero-shot Prompting

One-shot, implica que el modelo realiza una tarea con **un solo** ejemplo demostrativo, esto es útil cuando Zero-shot no es suficiente.

Prompt Engineering Guide: One-shot Prompting

GPT-3 Few-shot

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

Few-shot, es dar múltiples ejemplos demostrativos, en otras palabras es una generalización de one-shot.

GPT-3 ¿Sin Fine-tuning?

Fine-tuning, implica actualizar gradientes del modelo, la idea de GPT-3 es evitar manipular los gradientes para lograr generalizar en lugar de realizar tareas específicas.

Para GPT-3, lo mas importante es la flexibilidad y eficiencia al tener un modelo capaz de hacer distintas tareas “on the fly” solamente utilizando **prompts**.

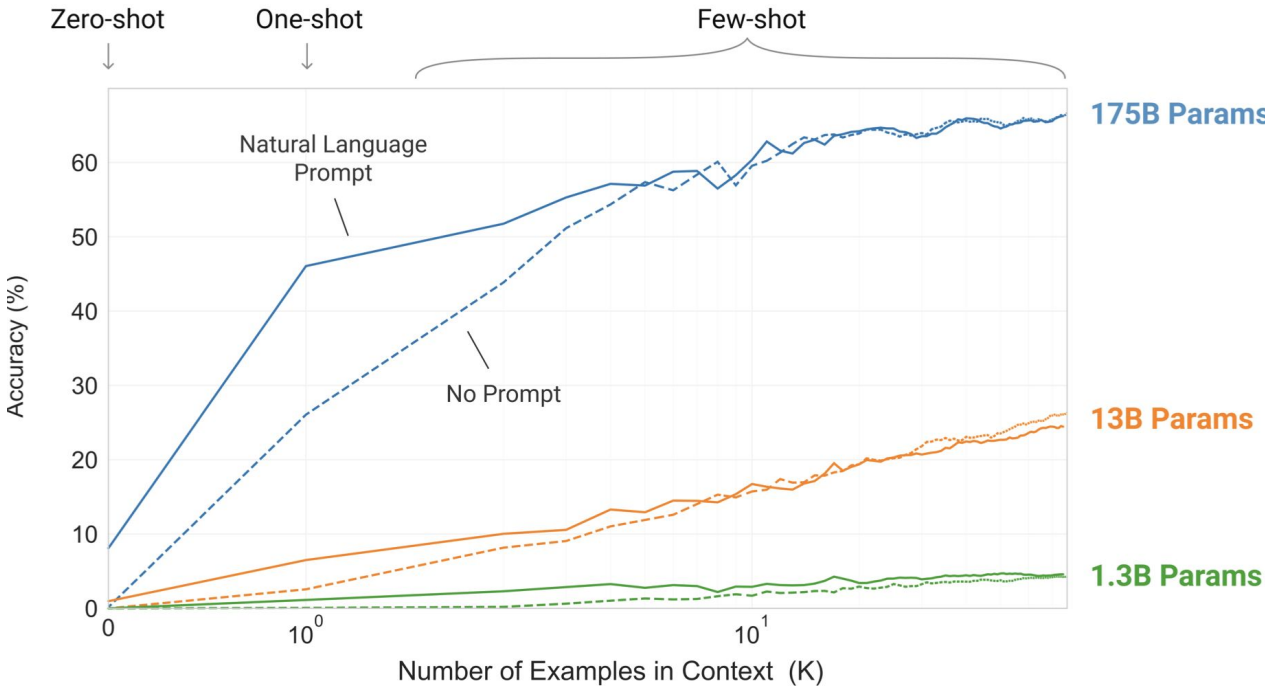
La razón por la cual GPT-3 es capaz de interpretar distintas tareas sin fine-tuning es debido a la diversidad de los datos. Por ejemplo, preguntas y respuestas en reddit, texto informativo de Wikipedia.

Los datasets utilizados cuentan con datos variados distintas tareas, idiomas, etc.

GPT-3 Explanation (Video)

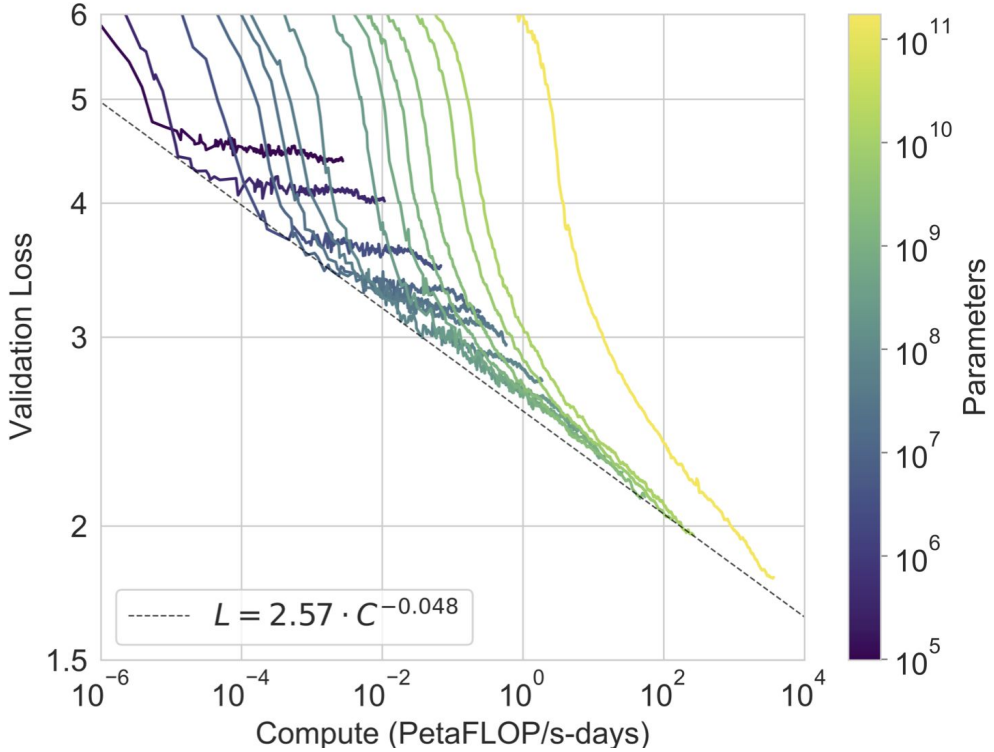
La manera de aprender de manera generalizada es llamado “in-context learning”

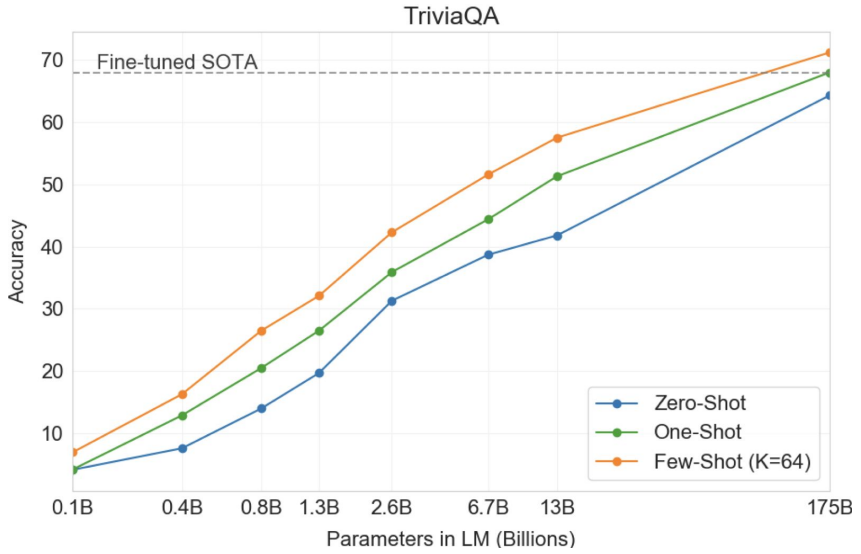
GPT-3 ¿Sin Fine-tuning?



Los modelos más grandes utilizan de manera más eficiente informacion “in-context”

GPT-3 ¿Sin Fine-tuning?





InstructGPT

GPT-3 demuestra que simplemente escalar una LLM permite realizar múltiples tareas, sin embargo esto no lo hace **explícitamente bueno dado instrucciones de usuarios.** Las LLMs pueden generar contenido tóxico, falso y no util para el usuario.

InstructGPT surge para corregir las debilidades de GPT-3 mediante **fine-tuning y reinforcement-learning,** la versión 1.3B InstructGPT contiene respuestas superiores a 175B GPT-3.

OpenAI InstructGPT

InstructGPT Review

InstructGPT

Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

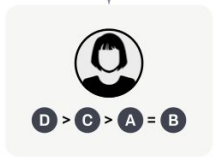
This data is used to fine-tune GPT-3 with supervised learning.

Step 2

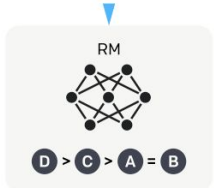
Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using reinforcement learning.

InstructGPT utiliza un reward model con los outputs preferidos por humanos, luego se optimiza el reward model mediante Proximal Policy Optimization (PPO).

HuggingFace Reinforcement Learning Course

Transformer Reinforcement Learning (TRL)

InstructGPT Dataset

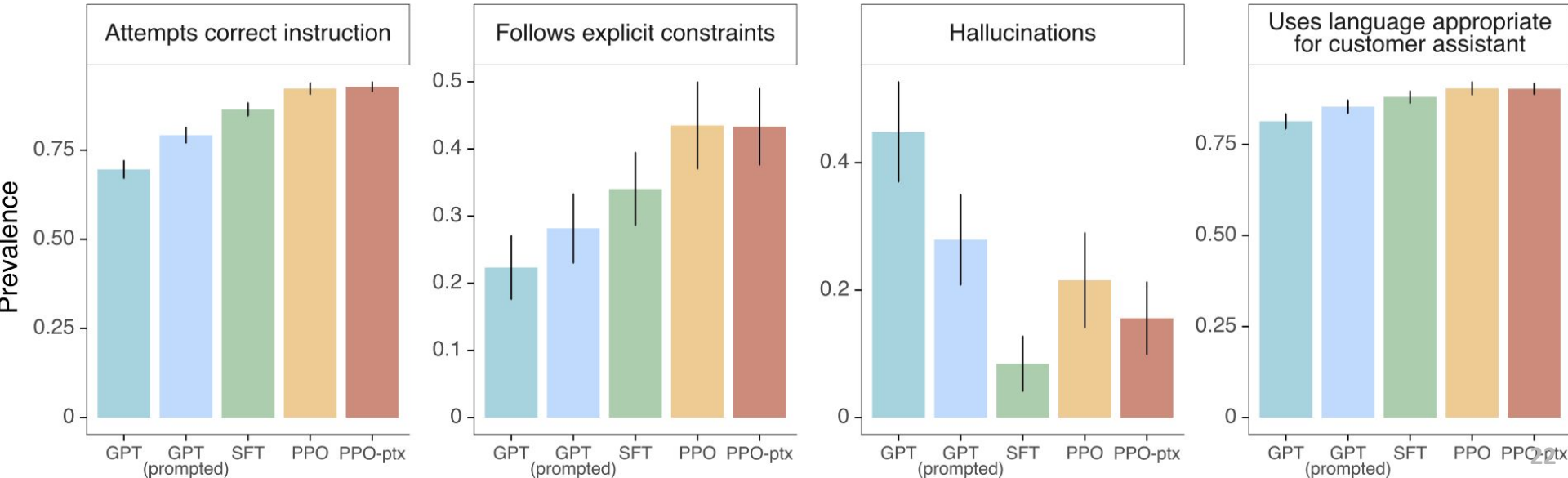
El dataset consiste en prompts ingresados al API de OpenAI mediante OpenAI Playground.

Un grupo de personas llamados “labelers” escribían prompts en texto crudo, Few-shot y basados en usuarios.

Table 6: Dataset sizes, in terms of number of prompts.

SFT Data			RM Data			PPO Data		
split	source	size	split	source	size	split	source	size
train	labeler	11,295	train	labeler	6,623	train	customer	31,144
train	customer	1,430	train	customer	26,584	valid	customer	16,185
valid	labeler	1,550	valid	labeler	3,488			
valid	customer	103	valid	customer	14,399			

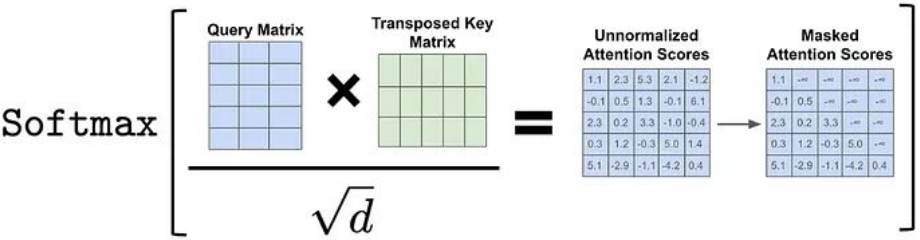
InstructGPT Resultados

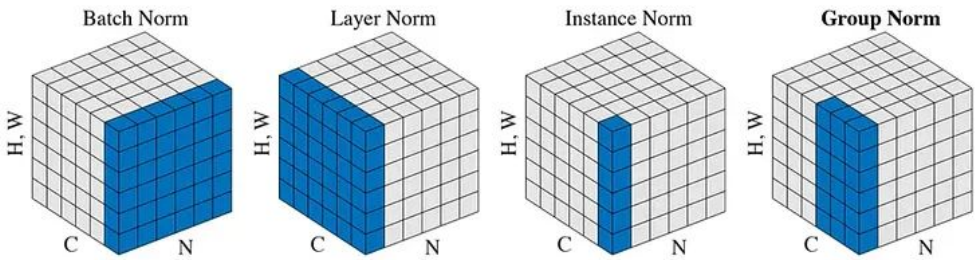


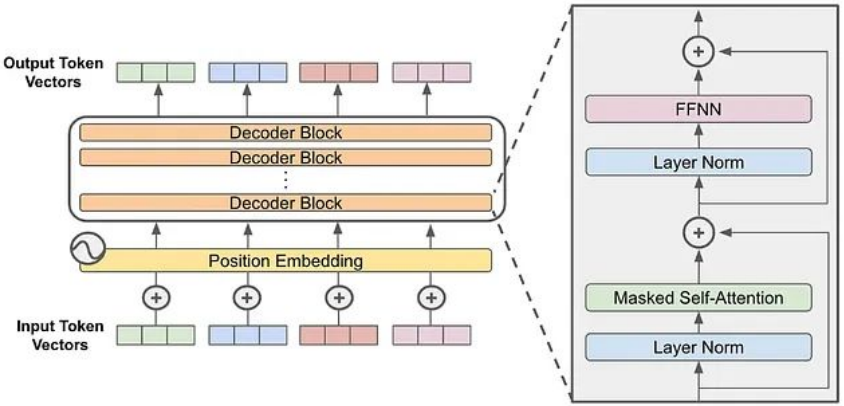
Transformers Decoders

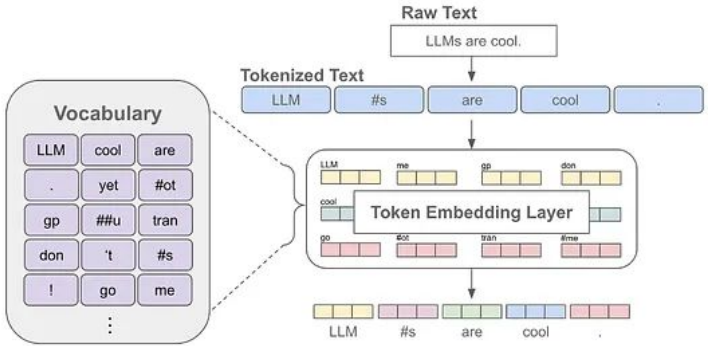
[Decoder-Only Transformers: The Workhorse of Generative LLMs.]

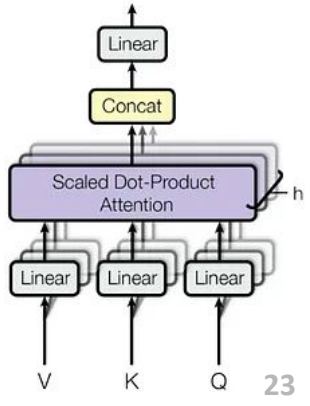
[LLM Visualization]











Transformer Explainer

Transformer Explainer es un UI con un GPT-2 ejecutado en el browser en tiempo real.

Paper

Repo

Apple Intelligence

Apple Intelligence: how to build an on-device LLM

Arquitecturas de LLMs

Algunas Arquitecturas del Estado del Arte

- Llama Family (Meta)
- Mistral (Mistral AI)
- Gemma (Google)
- Claude (Anthropic)
- Grok (xAI)
- DeepSeek (DeepSeek)

Chatbot Arena

Chatbot Arena es un Leaderboard con un sistema de ranking para evaluar LLMs.

The Rise and Rise of A.I.

Large Language Models (LLMs)

LLAMA

LLama, enfatiza el uso de datasets públicos, LLama 13B supera a GPT-3 en la mayoría de benchmarks a pesar de ser x10 más pequeño (similar a InstructGPT)

El dataset contiene **1.4T** de tokens.

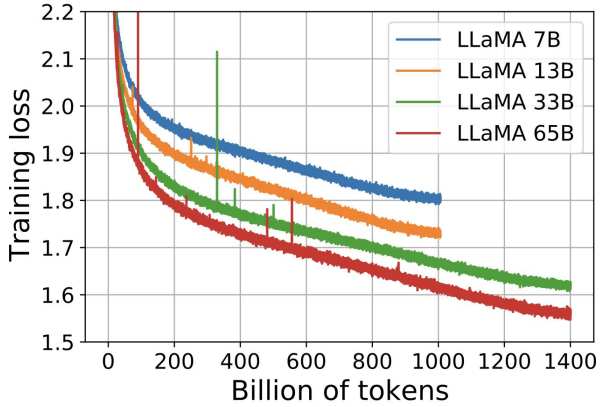
El context length es de 2048 tokens.

Dataset	Sampling prop.	Epochs	Disk size
CommonCrawl	67.0%	1.10	3.3 TB
C4	15.0%	1.06	783 GB
Github	4.5%	0.64	328 GB
Wikipedia	4.5%	2.45	83 GB
Books	4.5%	2.23	85 GB
ArXiv	2.5%	1.06	92 GB
StackExchange	2.0%	1.03	78 GB

LLAMA

La arquitectura es el Transformer Decoder, el cual fue adaptado con mejoras de arquitectras previas como:

- Capa de Pre-normalizacion (GPT-3) RMSNorm
- Función de activación SwiGLU (PaLM)
- Rotary Positional Embeddings RoPE (GPTNeo)
- Efficient Self-Attention



Fue entrenado con batches de 4M de tokens.

params	dimension	n heads	n layers	learning rate	batch size	n tokens
6.7B	4096	32	32	$3.0e^{-4}$	4M	1.0T
13.0B	5120	40	40	$3.0e^{-4}$	4M	1.0T
32.5B	6656	52	60	$1.5e^{-4}$	4M	1.4T
65.2B	8192	64	80	$1.5e^{-4}$	4M	1.4T

LLAMA

2048 GPUs NVIDIA A100-80GB por un periodo de aproximadamente 5 meses en desarrollar

2,638 MWh y un total de emisiones of 1,015 tCO2eq

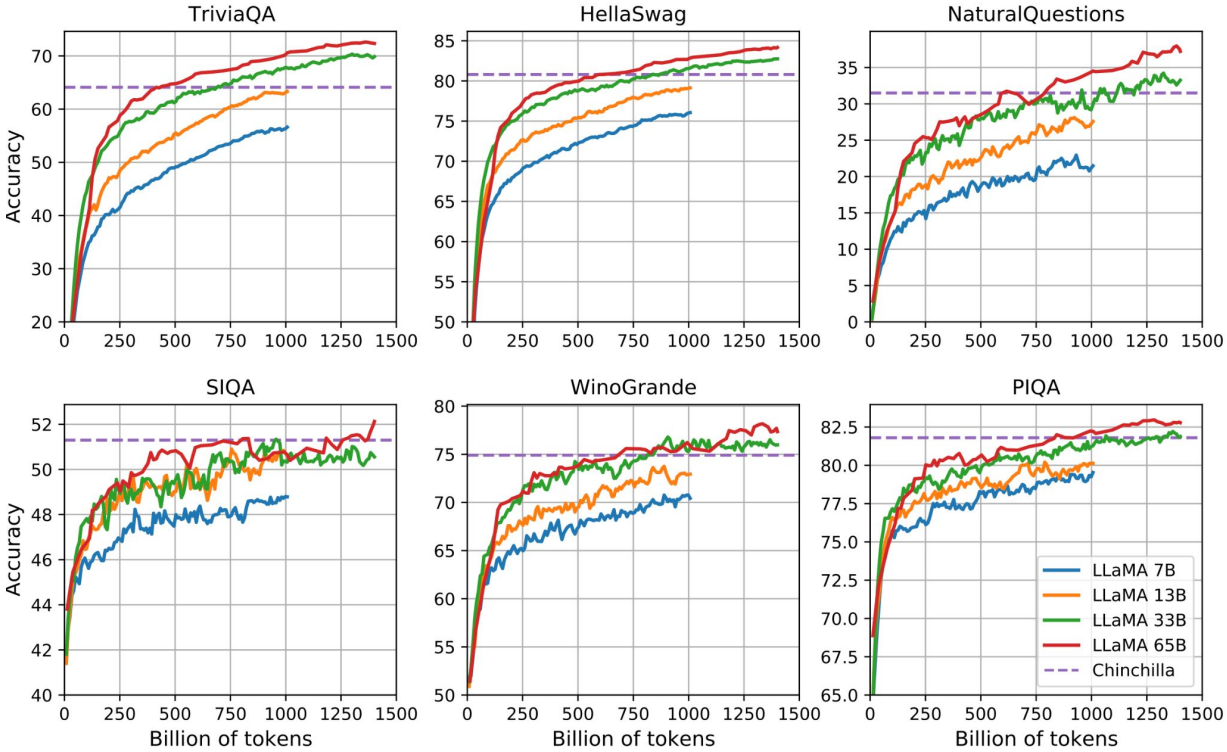
		BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA
GPT-3	175B	60.5	81.0	-	78.9	70.2	68.8	51.4	57.6
Gopher	280B	79.3	81.8	50.6	79.2	70.1	-	-	-
Chinchilla	70B	83.7	81.8	51.3	80.8	74.9	-	-	-
PaLM	62B	84.8	80.5	-	79.7	77.0	75.2	52.5	50.4
PaLM-cont	62B	83.9	81.4	-	80.6	77.0	-	-	-
PaLM	540B	**88.0**	82.3	-	83.4	**81.1**	76.6	53.0	53.4
LLaMA	7B	76.5	79.8	48.9	76.1	70.1	72.8	47.6	57.2
	13B	78.1	80.1	50.4	79.2	73.0	74.8	52.7	56.4
	33B	83.1	82.3	50.4	82.8	76.0	**80.0**	**57.8**	58.6
	65B	85.3	**82.8**	**52.3**	**84.2**	77.0	78.9	56.0	**60.2**

Table 3: **Zero-shot performance on Common Sense Reasoning tasks.**

		0-shot	1-shot	5-shot	64-shot
GPT-3	175B	14.6	23.0	-	29.9
Gopher	280B	10.1	-	24.5	28.2
Chinchilla	70B	16.6	-	31.5	35.5
PaLM	8B	8.4	10.6	-	14.6
	62B	18.1	26.5	-	27.6
	540B	21.2	29.3	-	39.6
LLaMA	7B	16.8	18.7	22.0	26.1
	13B	20.1	23.4	28.1	31.9
	33B	**24.9**	28.3	32.9	36.0
	65B	23.8	**31.0**	**35.0**	**39.9**

Table 4: **NaturalQuestions.** Exact match performance

LLAMA Resultados



LLAMA

Implementación Github

	GPU Type	GPU Power consumption	GPU-hours	Total power consumption	Carbon emitted (tCO_2eq)
OPT-175B	A100-80GB	400W	809,472	356 MWh	137
BLOOM-175B	A100-80GB	400W	1,082,880	475 MWh	183
LLaMA-7B	A100-80GB	400W	82,432	36 MWh	14
LLaMA-13B	A100-80GB	400W	135,168	59 MWh	23
LLaMA-33B	A100-80GB	400W	530,432	233 MWh	90
LLaMA-65B	A100-80GB	400W	1,022,362	449 MWh	173

LLAMA-2

LLAMA-2 Es exactamente la misma arquitectura que LLAMA I, las diferencias radican:

- Mas parametros.
- Dataset incrementado en un 40%.
- Se duplica el context length a 4096 tokens.
- Implementación de Grouped-query attention (GQA).

También se liberó una versión fine-tuned llamada LLAMA 2-Chat, optimizada para casos de chat.

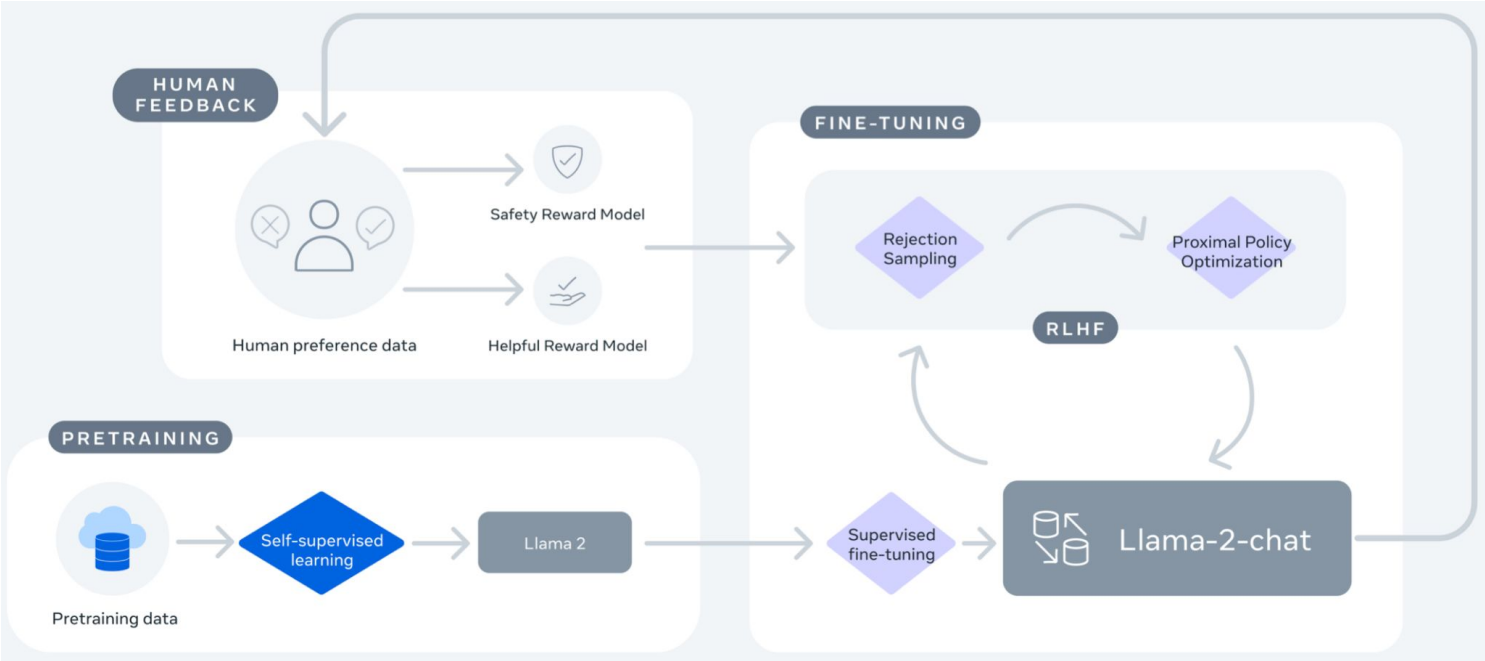
	Training Data	Params	Context Length	GQA	Tokens	LR
LLAMA 1	*See Touvron et al. (2023)*	7B	2k	✗	1.0T	3.0×10^{-4}
		13B	2k	✗	1.0T	3.0×10^{-4}
		33B	2k	✗	1.4T	1.5×10^{-4}
		65B	2k	✗	1.4T	1.5×10^{-4}
LLAMA 2	*A new mix of publicly available online data*	7B	4k	✗	2.0T	3.0×10^{-4}
		13B	4k	✗	2.0T	3.0×10^{-4}
		34B	4k	✓	2.0T	1.5×10^{-4}
		70B	4k	✓	2.0T	1.5×10^{-4}

LLAMA Tokenizer

Mismo tokenizer que Llama I (BPE) usando bytes para descomponer caracteres UTF-8 desconocidos. El vocabulary size is **32k tokens.**

LLAMA 2-Chat

Similar a InstructGPT, se realiza SFT y RL con feedback humano.



LLAMA 2 Carbon Footprint

		Time (GPU hours)	Power Consumption (W)	Carbon Emitted (tCO_2eq)
LLAMA 2	7B	184320	400	31.22
	13B	368640	400	62.44
	34B	1038336	350	153.90
	70B	1720320	400	291.42
Total		3311616		539.00

LLAMA 2

Benchmark (shots)	GPT-3.5	GPT-4	PaLM	PaLM-2-L	Llama 2
MMLU (5-shot)	70.0	**86.4**	69.3	78.3	68.9
TriviaQA (1-shot)	–	–	81.4	**86.1**	85.0
Natural Questions (1-shot)	–	–	29.3	**37.5**	33.0
GSM8K (8-shot)	57.1	**92.0**	56.5	80.7	56.8
HumanEval (0-shot)	48.1	**67.0**	26.2	–	29.9
BIG-Bench Hard (3-shot)	–	–	52.3	**65.7**	51.2

Model	Size	Code	Commonsense Reasoning	World Knowledge	Reading Comprehension	Math	MMLU	BBH	AGI Eva
MPT	7B	20.5	57.4	41.0	57.5	4.9	26.8	31.0	23.5
	30B	28.9	64.9	50.0	64.7	9.1	46.9	38.0	33.8
Falcon	7B	5.6	56.1	42.8	36.0	4.6	26.2	28.0	21.2
	40B	15.2	69.2	56.7	65.7	12.6	55.4	37.1	37.0
Llama 1	7B	14.1	60.8	46.2	58.5	6.95	35.1	30.3	23.9
	13B	18.9	66.1	52.6	62.3	10.9	46.9	37.0	33.9
	33B	26.0	70.0	58.4	67.6	21.4	57.8	39.8	41.7
	65B	30.7	70.7	60.5	68.6	30.8	63.4	43.5	47.6
Llama 2	7B	16.8	63.9	48.9	61.3	14.6	45.3	32.6	29.3
	13B	24.5	66.9	55.4	65.8	28.7	54.8	39.4	39.1
	34B	27.8	69.9	58.7	68.0	24.2	62.6	44.1	43.4
	70B	**37.5**	**71.9**	**63.6**	**69.4**	**35.2**	**68.9**	**51.2**	**54.2**

Mistral

En 2023, MistralAI lanzo Mistral 7B, el cual supera a LLAMA 2 13B en todos los benchmarks.

La arquitectura es un Decoder similar a LLAMA y GPT.

Su dataset es closed source.

Implementa:

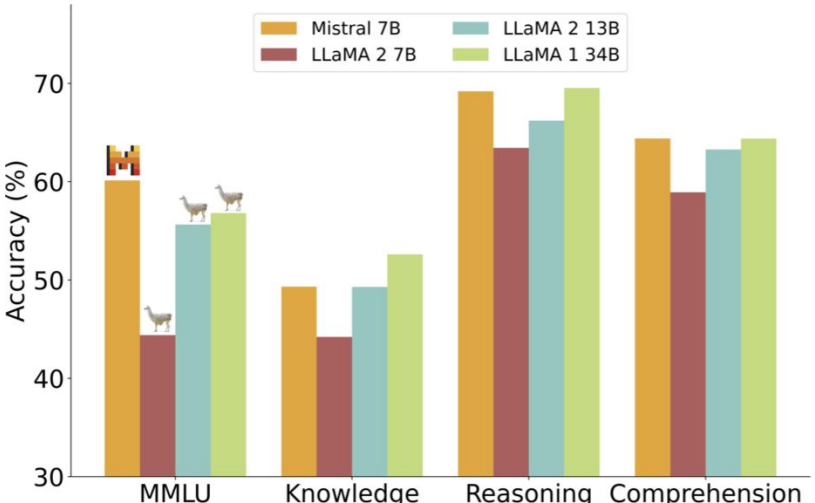
- Sliding Window Attention
- Grouped Query Attention (inference)
- Rolling Buffer Cache

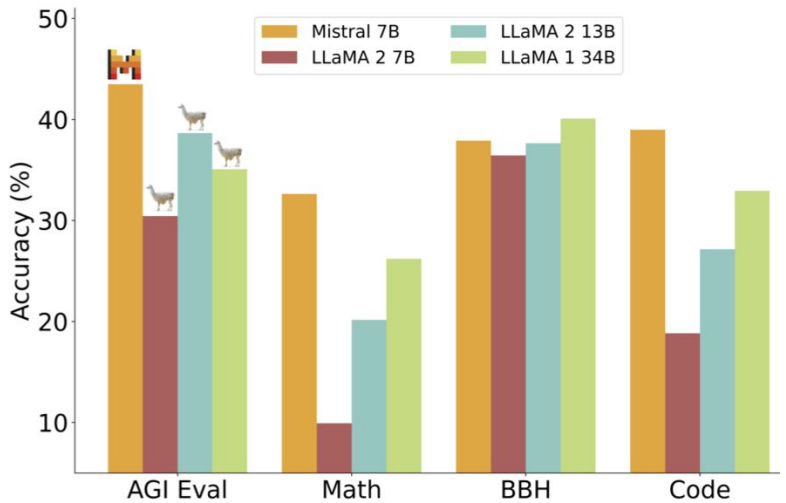
Parameter	Value
dim	4096
n_layers	32
head_dim	128
hidden_dim	14336
n_heads	32
n_kv_heads	8
window_size	4096
context_len	8192
vocab_size	32000

Table 1: Model architecture.

Mistral

Model	Modality	MMLU	HellaSwag	WinoG	PIQA	Arc-e	Arc-c	NQ	TriviaQA	HumanEval	MBPP	MATH	GSM8K
LLaMA 2 7B	Pretrained	44.4%	77.1%	69.5%	77.9%	68.7%	43.2%	24.7%	63.8%	11.6%	26.1%	3.9%	16.0%
LLaMA 2 13B	Pretrained	55.6%	**80.7%**	72.9%	80.8%	75.2%	48.8%	**29.0%**	**69.6%**	18.9%	35.4%	6.0%	34.3%
Code-Llama 7B	Finetuned	36.9%	62.9%	62.3%	72.8%	59.4%	34.5%	11.0%	34.9%	**31.1%**	**52.5%**	5.2%	20.8%
Mistral 7B	Pretrained	**60.1%**	**81.3%**	**75.3%**	**83.0%**	**80.0%**	**55.5%**	**28.8%**	**69.9%**	**30.5%**	47.5%	**13.1%**	**52.2%**





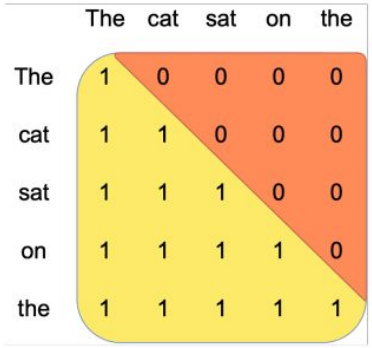
Sliding-Window Attention

Variante de Attention presentada en 2020 bajo el paper de Longformer con O(N) en lugar de $O(N^2)$.

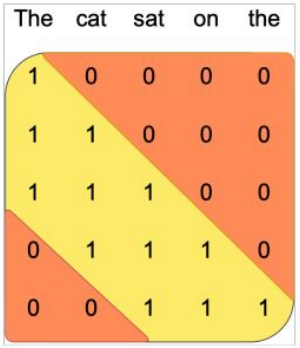
Consiste en focalizar en una ventana de tamaño fijo dentro de la matriz de atención.

Sliding Window Attention Explained

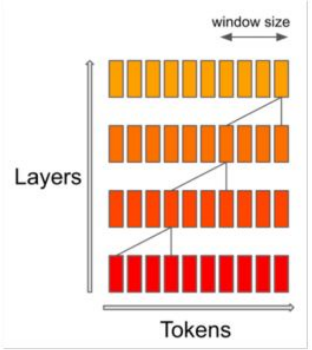
SWA Pytorch Implementation



Vanilla Attention



Sliding Window Attention



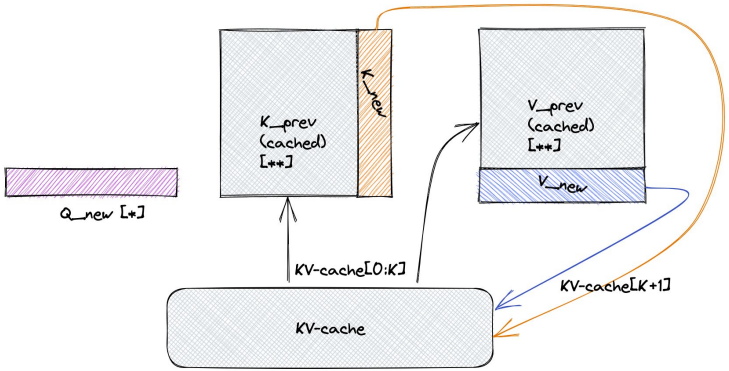
Effective Context Length

KV-Cache

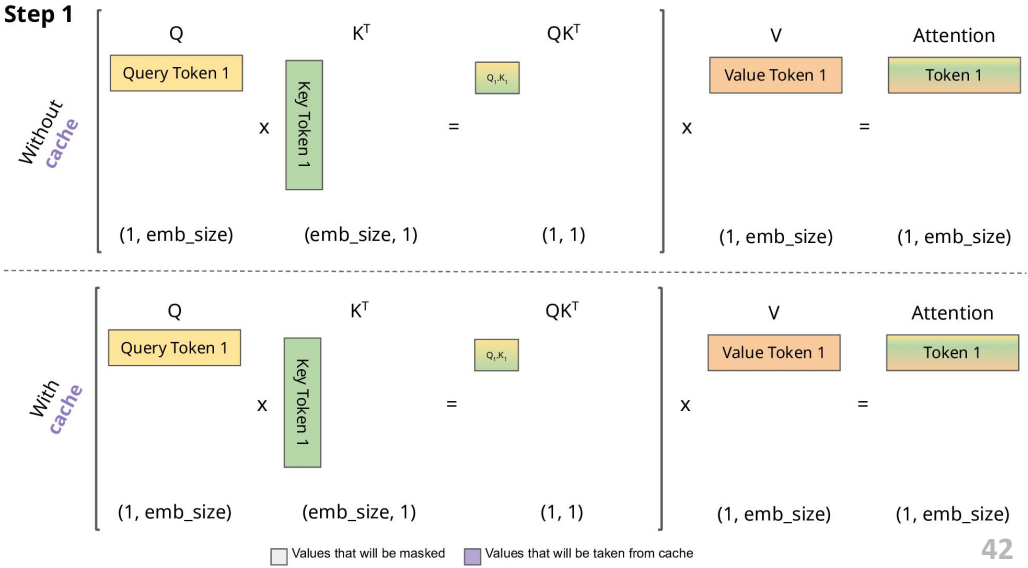
[KV caching a deeper look]
[What is KV-cache?]

Una LLM genera un token durante cada forward pass, esto implica calcular attention en cada forward, haciendo que la inferencia sea lenta.

Al hacer caching de las matrices de keys y values, esto solamente implicaría agregar el nuevo vector calculado, haciendo que la inferencia incremente considerablemente.

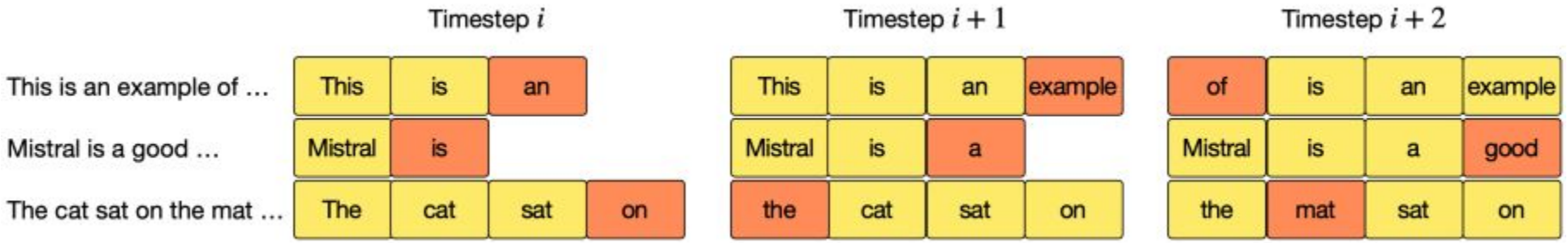


Notes:
---
* When processing token[K], we only need the K'th row of Q
** When processing token[K], we require the full K & V tensors, but we can mostly reuse the cached values (This enables skipping the computation of K & V



Rolling Buffer Cache

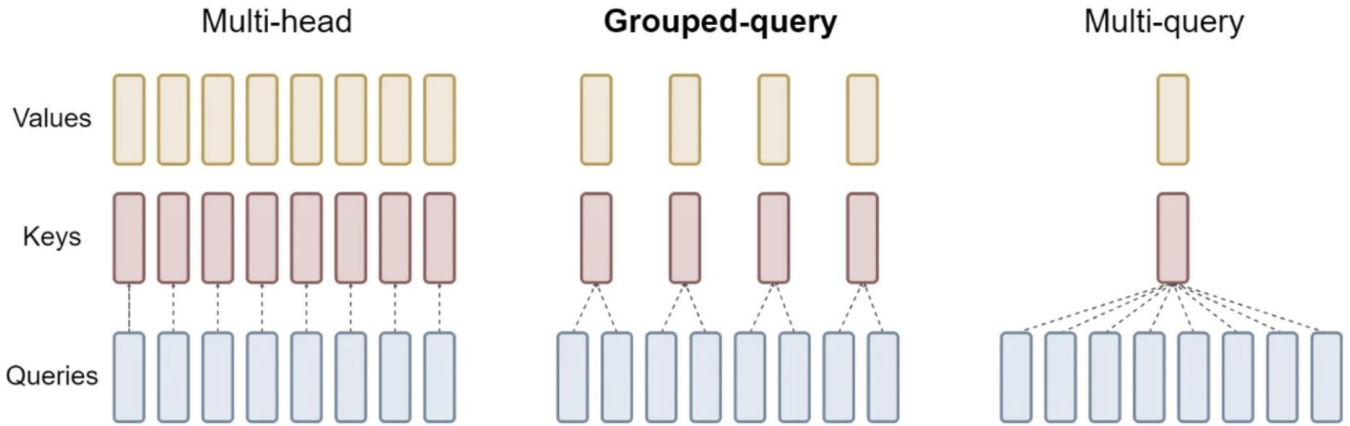
Mistral propone un caché dinámico de dimensiones fijas, lo cual reduce considerablemente el uso de memoria



Grouped-Query Attention

[Guia Explicativa]

[Demystifying GQA]



¿En qué difieren las arquitecturas?

El mayor **cuello de botella** en Transformers es el mecanismo de atención!

La mayoría de diferencias se encuentran en:

- Cambios de Attention.
- Context Length.
- # de parámetros.
- Datasets
- Positional Embeddings.

Preguntas?