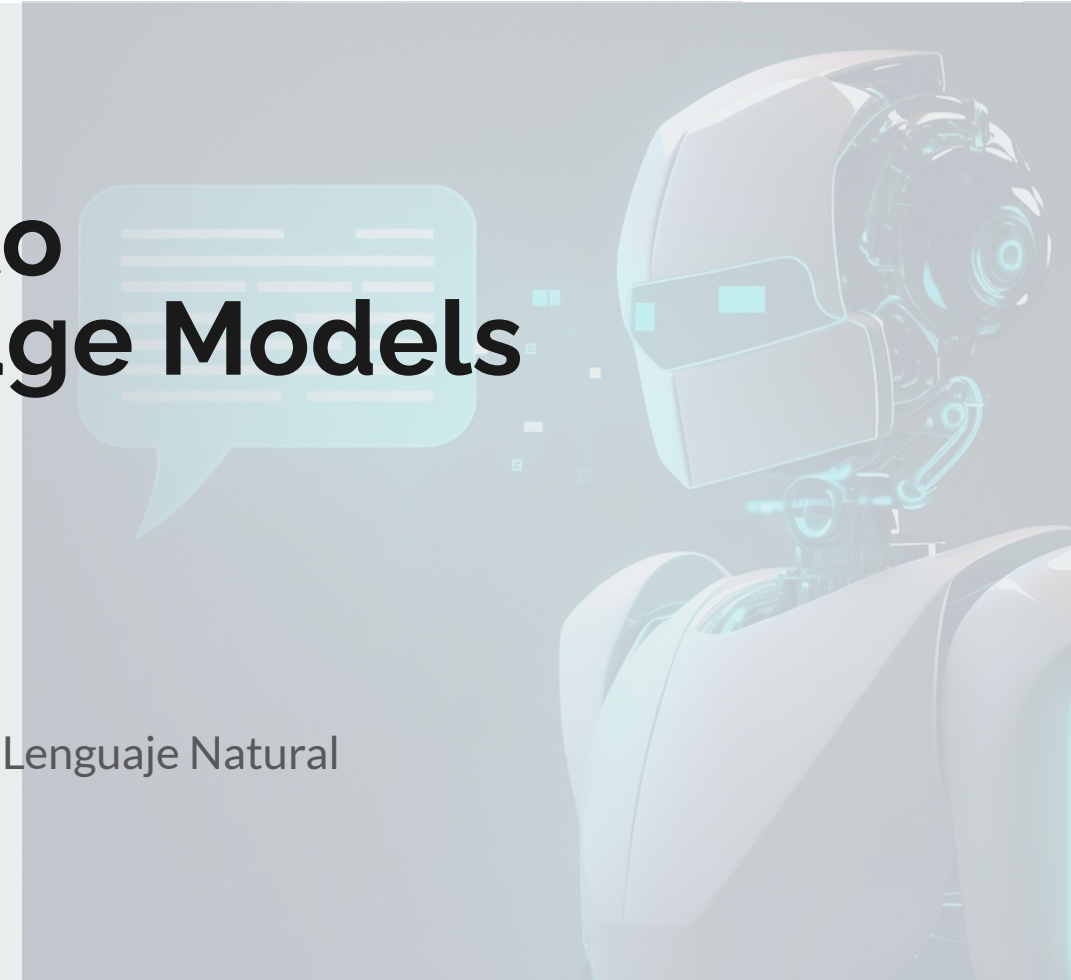
Introduction to Large Language Models and Agents



Ronald Cardenas Acosta, Ph.D.
Investigador en Procesamiento de Lenguaje Natural

Syllabus

Unidad 1	Conceptos Básicos
Unidad 2	Modelado de Lenguaje
Unidad 3	Pre-entrenamiento y Fine-tuning
Unidad 4	**Post-entrenamiento, aplicaciones de LLMs**

Unidad 4: Post-Entrenamiento y aplicaciones de LLMs

Metodos de Generacion

Post-training & Preference Optimizacion

Entrenamiento Eficiente de LLMs

Interfaces de Inferencia

Retrieval Augmented Generation

Metodos de Generacion

Metodos de Generacion

- **Metodos de Busqueda**
  - Greedy Search (Busqueda Avara)
  - Beam Search

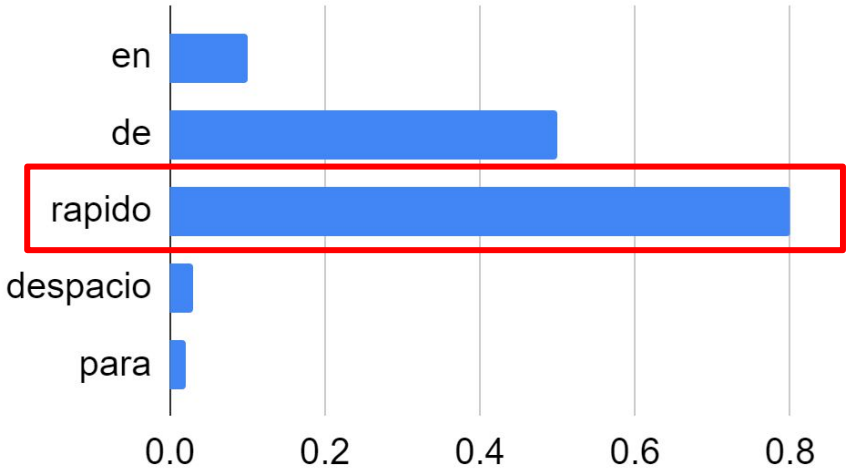
- **Técnicas de muestreo**
  - Muestreo ancestral
  - Seleccion Top-K
  - Muestro Nuclear

Greedy Search (Busqueda Avara)

- Selecciona la palabra con probabilidad **más alta**

“Me gusta correr ___”

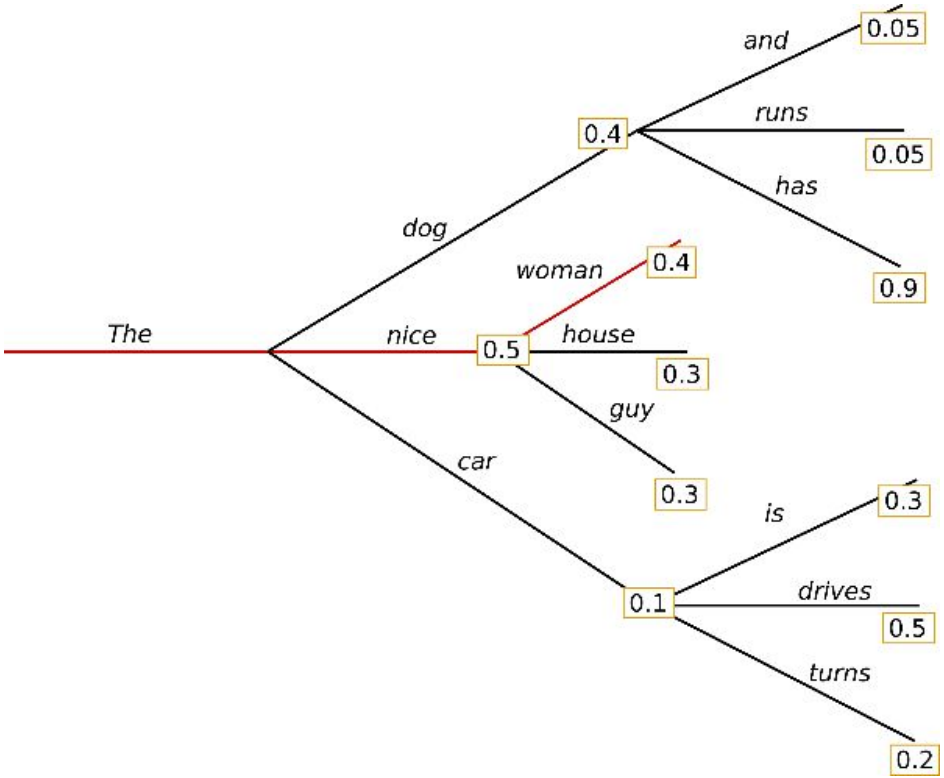
P(* | me,gusta,correr) :



Greedy Search (Busqueda Avara)

- Selecciona la palabra con probabilidad **más alta**
- **Desventaja:**
  - La secuencia final puede no ser la más probable

Greedy Search (Busqueda Avara)



Greedy Search (Busqueda Avara)

- Selecciona la palabra con probabilidad **más alta**
- **Desventaja:**
  - La secuencia final puede no ser la más probable
  - Tiende a producir texto repetitivo

Beam Search (Busqueda en Pila)

- En cada paso, mantiene las **N** secuencias más probables en una pila (stack)

N=2

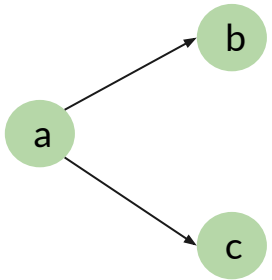
a

Iteracion 0
Beam = {a}

Beam Search (Busqueda en Pila)

- En cada paso, mantiene las **N** secuencias más probables en una pila (stack)

N=2



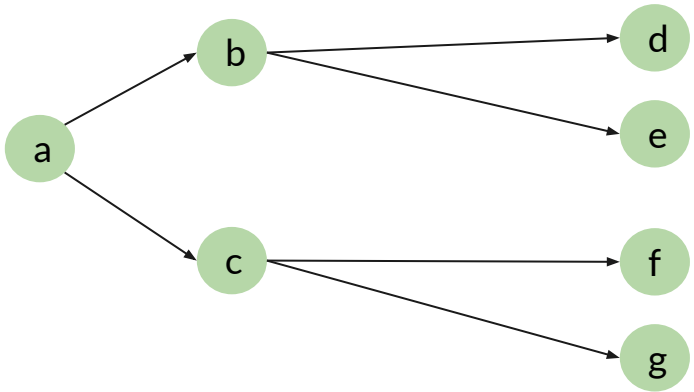
Iteracion 0
Beam = {a}

Iteracion 1
Beam = {ab, ac}

Beam Search (Busqueda en Pila)

- En cada paso, mantiene las **N** secuencias más probables en una pila (stack)

N=2



p(abd) = 0.2

p(abe) = 0.8

p(acf) = 0.6

p(acg) = 0.4

Iteracion 0
Beam = {a}

Iteracion 1
Beam = {ab, ac}

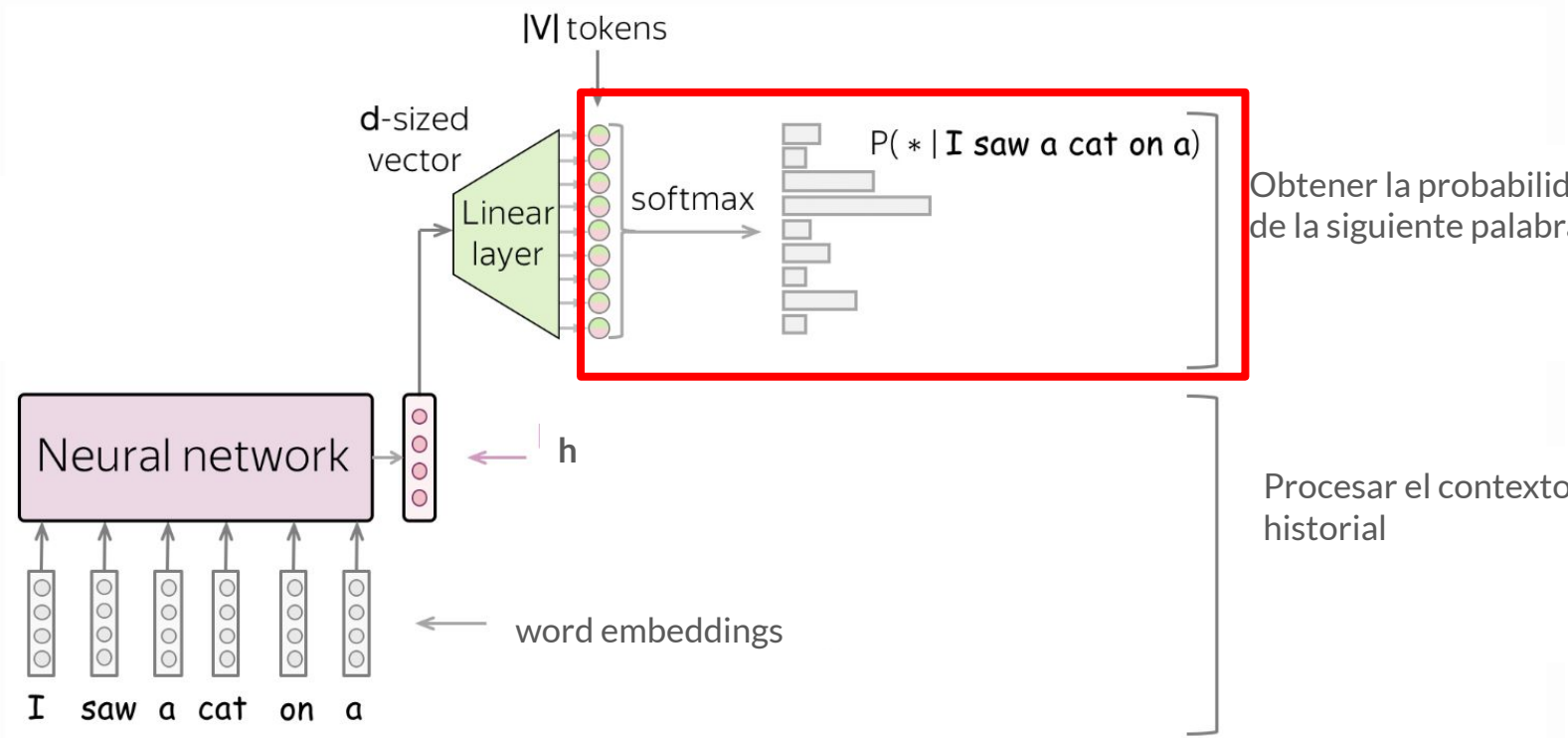
Iteracion 2
Beam = {abe, acf}

Beam Search (Busqueda en Pila)

- En cada paso, mantiene las **N** secuencias más probables en una pila (stack)
- **Ventajas**
  - Secuencias con mayor probabilidad que Greedy Search
  - Texto es más fluido, coherente
- **Desventajas**
  - Las N secuencias pueden no mostrar mucha diversidad léxica
  - Mientras mas alto N, mas costoso computacionalmente.

Generación de Texto: Métodos de Muestreo

Modelos Neurales de Lenguaje



Modelos Neurales de Lenguaje

- La probabilidad de token **a_i** se define como

$$p(a_i|a_{<i}) = softmax(l_i) = \frac{exp(l_i)}{\sum_{j \in V} exp(l_j)}$$

  - **L_i**: score dado a **a_i** por el modelo de lenguaje, aka **logit**

Modelos Neurales de Lenguaje

- La probabilidad de token **a_i** se define como

$$p(a_i|a_{<i}) = softmax(l_i) = \frac{exp(l_i)}{\sum_{j \in V} exp(l_j)}$$

L_i: score dado a **a_i** por el modelo de lenguaje, aka **logit**

- **Desventaja**
  - La distribución puede ser muy desbalanceada

Modelos Neurales de Lenguaje

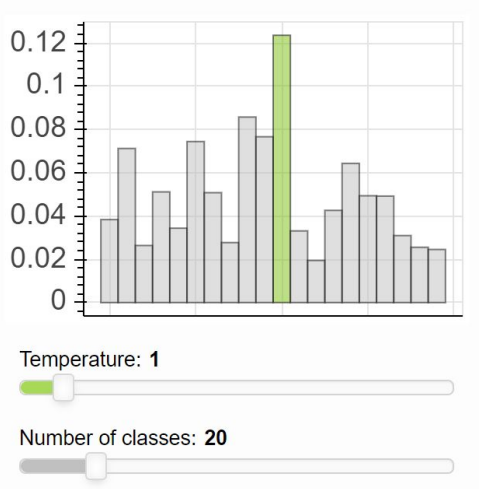
- Desventaja
  - La distribución puede ser muy desbalanceada
  - Solucion: re-escalar la distribucion
- Temperatura tau

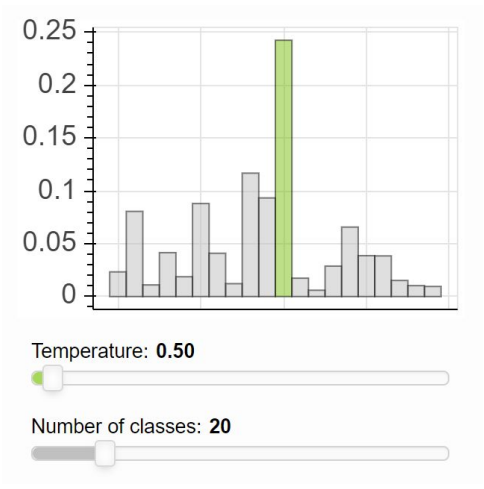
$$p(a_i|a_{<i}) = softmax(l_i/\tau) = \frac{exp(\frac{l_i}{\tau})}{\sum_{j \in V} exp(\frac{l_j}{\tau})}$$

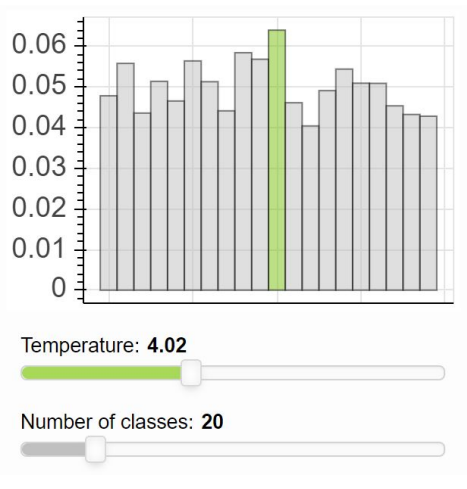
Modelos Neurales de Lenguaje

- Temperatura tau

$$p(a_i|a_{<i}) = softmax(l_i/\tau) = \frac{exp(\frac{l_i}{\tau})}{\sum_{j \in V} exp(\frac{l_j}{\tau})}$$







Ancestral Sampling (Muestro Ancestral)

- En cada paso, muestrea una palabra de nuestro modelo de lenguaje

$$a_i \sim P(*|a_1, \ldots, a_{i-1})$$

- Usualmente usado con re-escalamiento de temperatura
  - Bajo **tau**:
    - menos diversidad léxica
    - mas repetitivo
    - menos fluido
  - Alto **tau**:
    - mas fluido
    - más diversidad léxica
    - Mas chance de generar texto no veraz

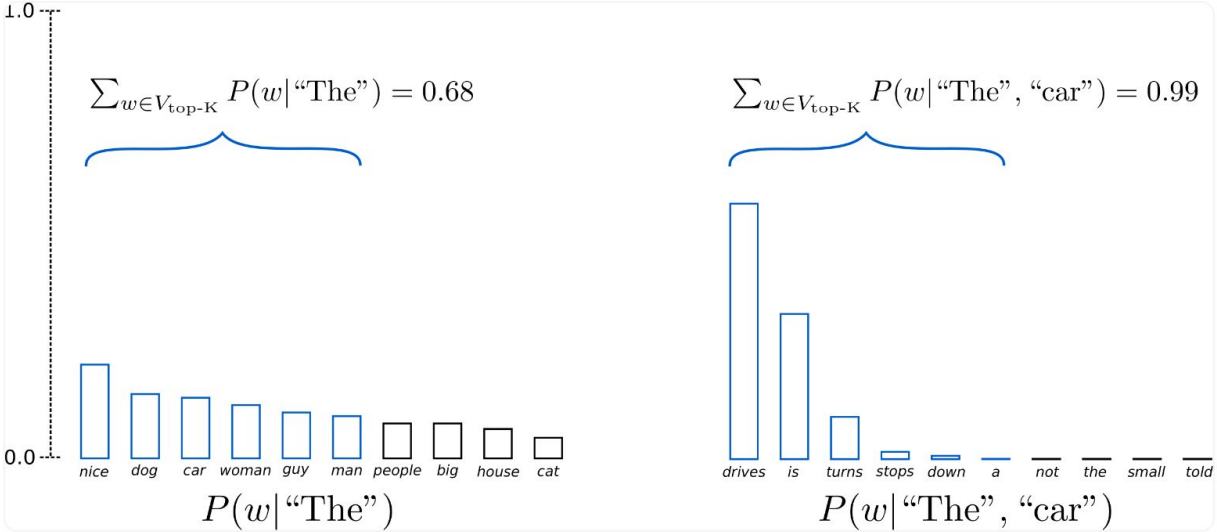
Top-K Sampling

- Antes de muestrear, limita las posibles palabras a las **K** con más probabilidad
  - V -> V_top-k

$$p_{top-k}(a_i|a_{<i}) = \frac{exp(l_i/\tau)}{\sum_{j \in V_{top-k}} exp(l_j/\tau)}$$

Top-K Sampling

- Antes de muestrear, limita las posibles palabras a las **K** con más probabilidad
  - V -> V_top-k



K=6

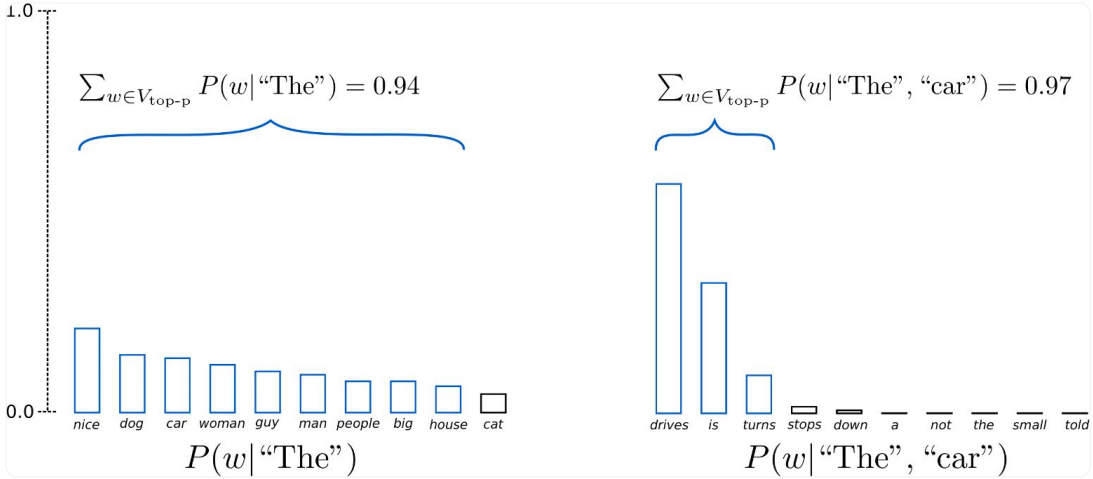
Nucleus Sampling

- Antes de muestrear, limita las posibles palabras al *conjunto que acumule* ***p*** *probabilidad*
  - V -> V_top-p

$$p_{top-p}(a_i|a_{<i}) = \frac{exp(l_i/\tau)}{\sum_{j \in V_{top-p}} exp(l_j/\tau)}$$

Nucleus Sampling

- Antes de muestrear, limita las posibles palabras al *conjunto que acumule* ***p*** *probabilidad*
  - V -> V_top-p



p=0.90

Generando con Texto: Demo

https://colab.research.google.com/drive/1jxlEdjJRXRcKyMWohRznvcrMvS25rCPg?usp=drive_link

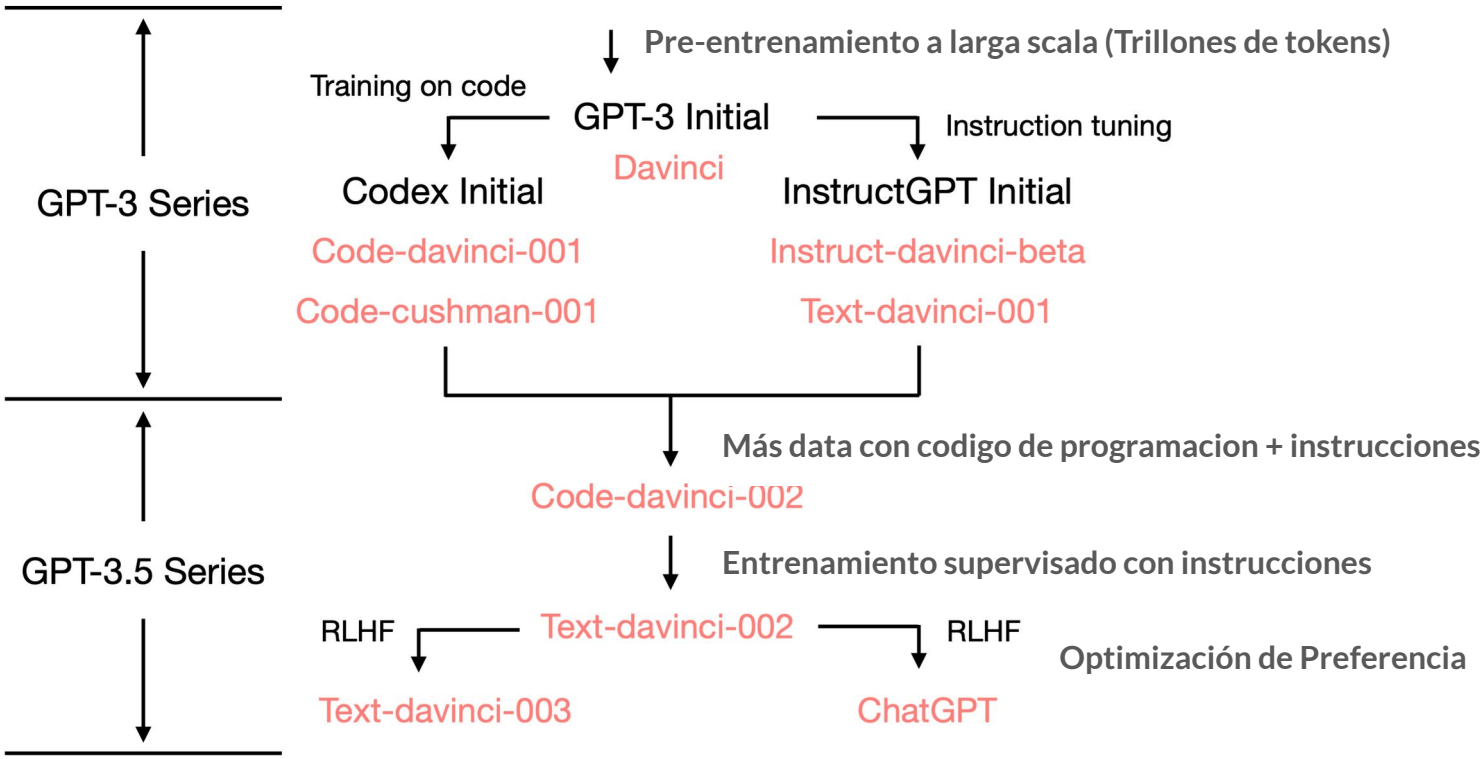
Post-entrenamiento

Fases de entrenamiento de una LLM

Entrenamiento por Instrucción

Optimización de Preferencia (Humana)

Como entrenar tu LLM: caso de OpenAI



Instruction Tuning: Siguiendo Instrucciones

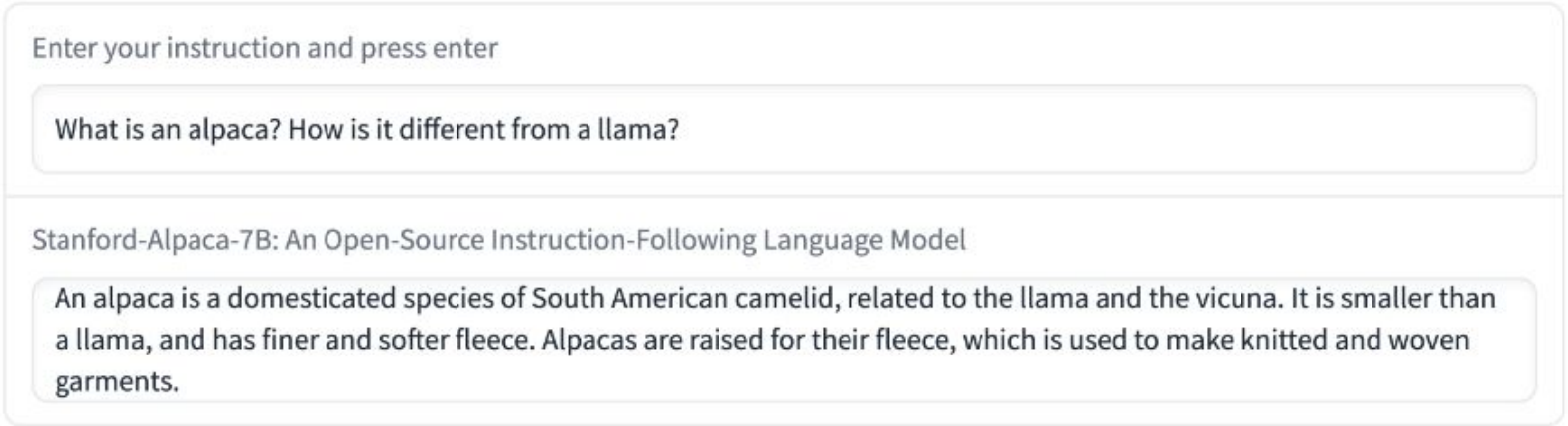
- **Instruccion:** Descripción básica de una tarea
  - *“Clasifica este texto en positivo o negativo”*
  - *“Resume este texto en una oración”*

- Modelos de Lenguaje pueden ser entrenados para seguir instrucciones
  - Supervised Fine-tuning sobre pares de instrucción - respuesta
  - Tarea sigue siendo ***next-token-prediction***

...

Instruction Tuning: ejemplo

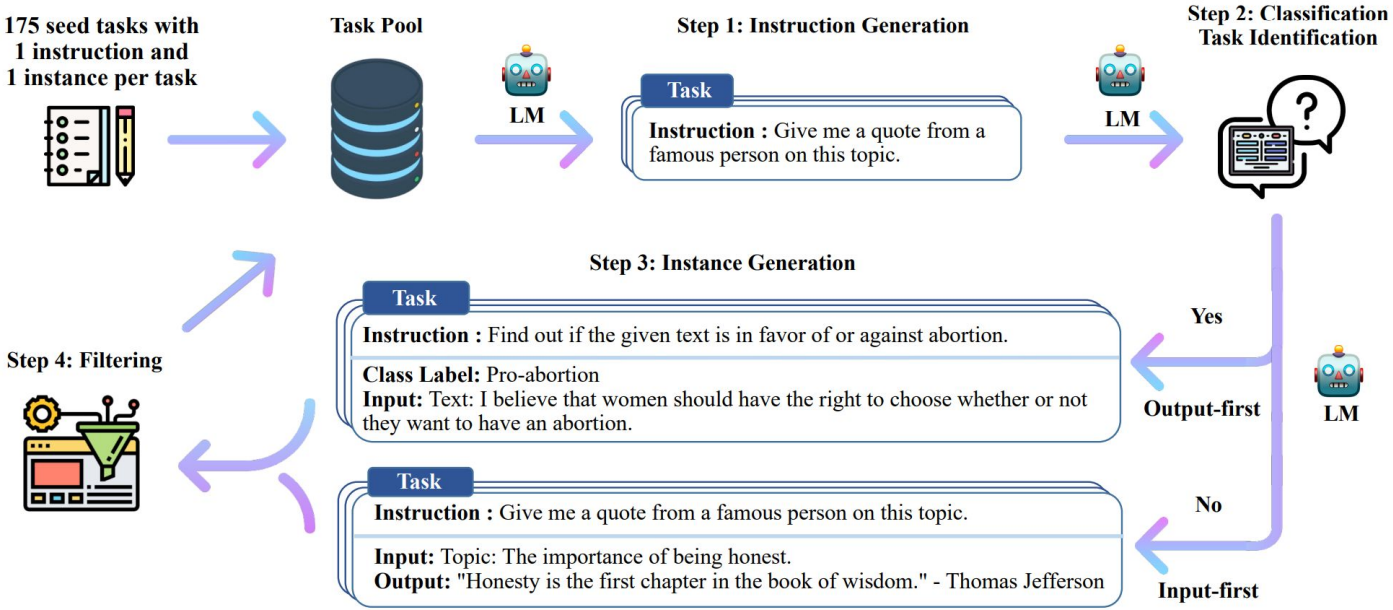
LLaMa + Instrucciones



Dataset de Instrucciones

Self-Instruct
https://arxiv.org/pdf/2212.10560

Dataset de Instrucciones



Self-Instruct
https://arxiv.org/pdf/2212.10560

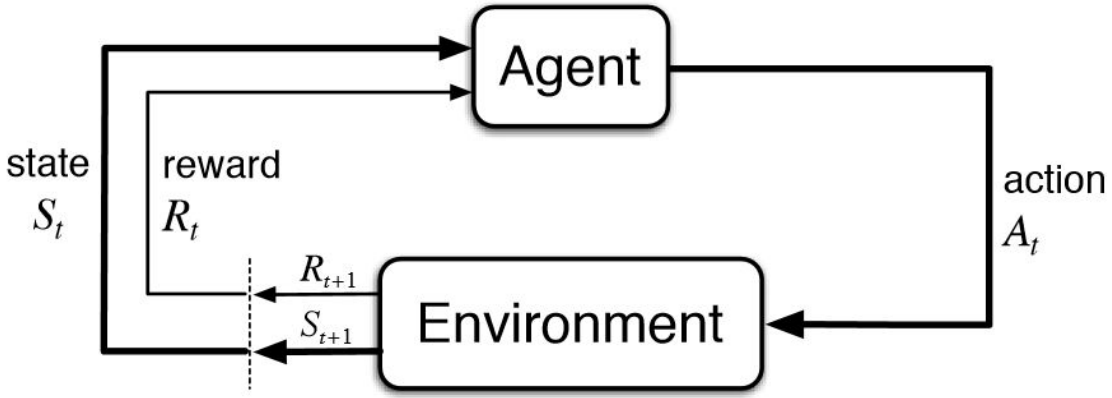
Optimización de Preferencia Humana

Aprendizaje Reforzado por Feedback Humano
(Reinforcement Learning from Human Feedback)

Direct Preference Optimization

Reinforcement Learning from Human Feedback

- Consiste en entrenar un LM para generar respuestas que sean más *útiles* para un usuario humano



Agent: LM
Env.: usuario
Reward: Score de preferencia
- reward alto -> mas preferencia

Estado S
Historial de conversación

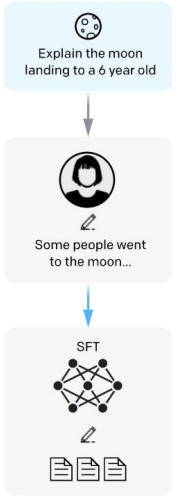
Reinforcement Learning from Human Feedback

Paso 1
Recolectar demostraciones y Entrenar un modelo policy supervisado

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.

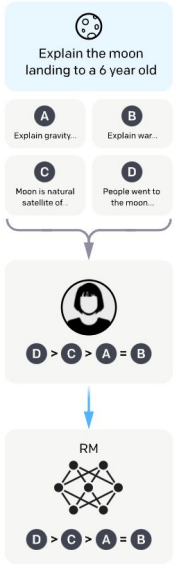


Paso 2
Recolectar data comparando generaciones y entrenar un modelo Reward

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.



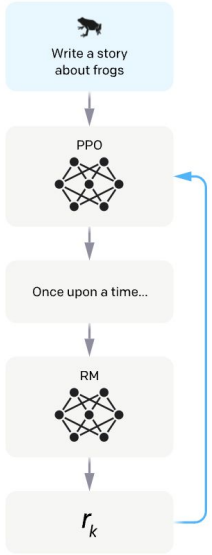
Paso 3
Optimizar un policy con el Reward usando reinforcement learning

A new prompt is sampled from the dataset.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.



Reinforcement Learning from Human Feedback

Proximal Policy Optimization Algorithms

John Schulman, Filip Wol

{joschu, filip

Direct Preference Optimization:
Your Language Model is Secretly a Reward Model

Rafael Rafailov[*†] **Archit Sharma**[*†] **Eric Mitchell**[*†]

Stefano Ermon[†‡] **Christopher D. Mannin**

[†]Stanford University [‡]CZ Biol
{rafailov,architsh,eric.mitchell}@c

KTO

SePO

SparsePO

RLOO

GRPO

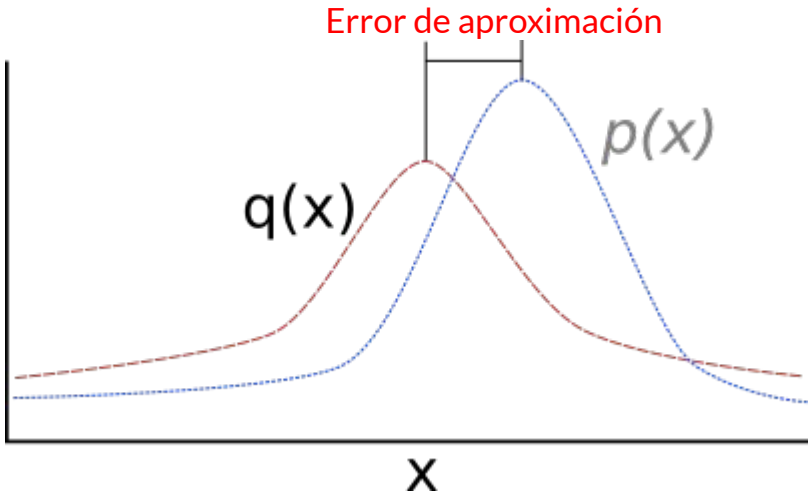
DAPO

....

Recap: Divergencia de Kullback-Leibler: Intuicion

- **p(x)**: distribución de probabilidad que queremos aproximar
  - Descripción completa desconocida
  - Solo tenemos muestras

- Si utilizamos distribución **q(x)** como substituta de p(x)
  - Cómo cuantificamos el error cometido?



Recap: Divergencia de Kullback-Leibler

- Medida de disimilaridad entre dos distribuciones de probabilidad

$$\mathbb{KL}(p||q) = \sum_{x \in \chi} p(x) \log \frac{p(x)}{q(x)}$$

$$\begin{aligned} \mathbb{KL}(p||q) &= \sum_{x \in \chi} p(x) \log p(x) - \sum_{x \in \chi} p(x) \log q(x) \\ &= -H(p) + H(p, q) \end{aligned}$$

- H(p,q) : cross-entropia
  - “Cantidad de bits necesarios para codificar *p* si usamos *q*”

- *“Distancia* KL”, “Entropia Relativa”
  - Estrictamente no es medida de distancia: KL(p||q) != KL(q||p)

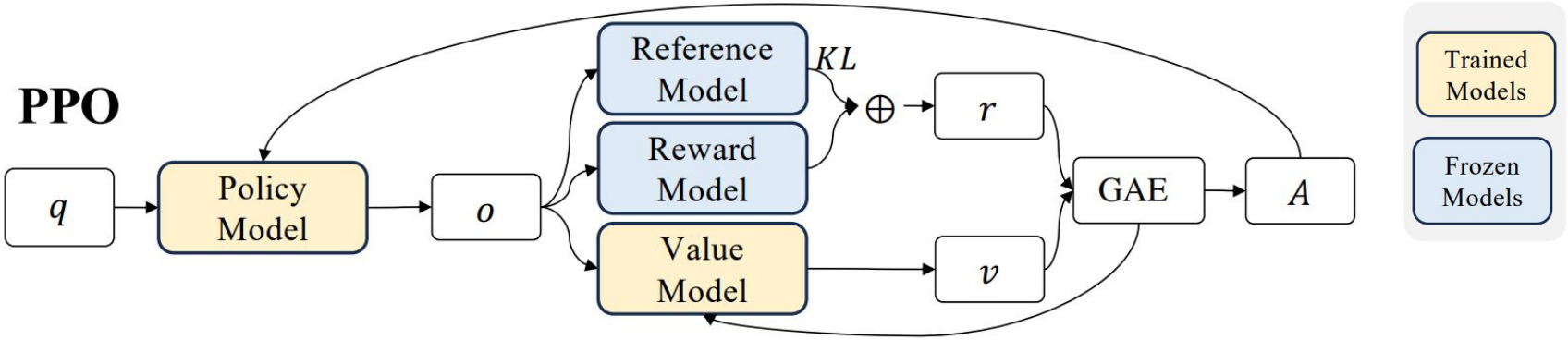
Policy Gradient Optimization

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta(a_t \mid s_t) \hat{A}_t \right].$$

π : policy (LLM)
a_t: accion en paso t
s_t: estado del
A_t: *advantage function*
estima la el posible beneficio relativo de usar acción ***a_t*** en vez de cualquier otra acción

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Preference Policy Optimizacion



GAE: grouped advantage estimation

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Preference Policy Optimizacion

$$\underset{\theta}{\text{maximize}} \quad \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \hat{A}_t \right]$$

$$\text{subject to} \quad \hat{\mathbb{E}}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta.$$

Policy a optimizar

Policy de referencia (fijo / frozen)

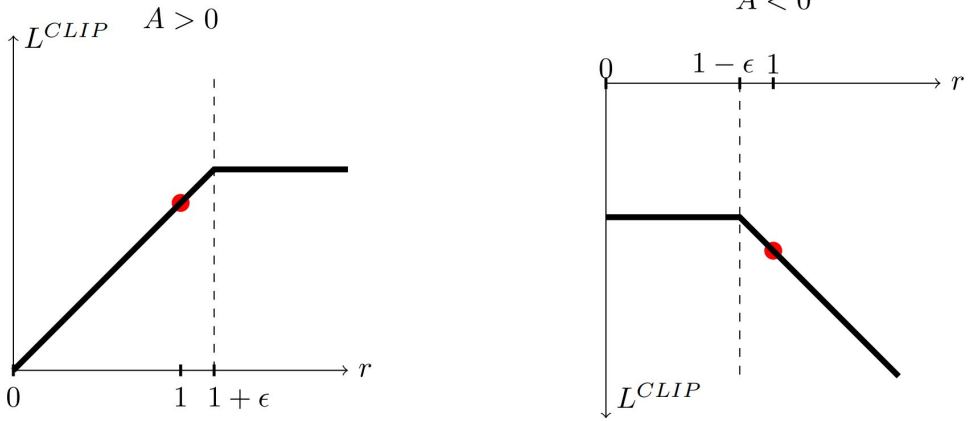
Restriccion en la divergencia, “distancia” de ambas distribuciones

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Preference Policy Optimizacion

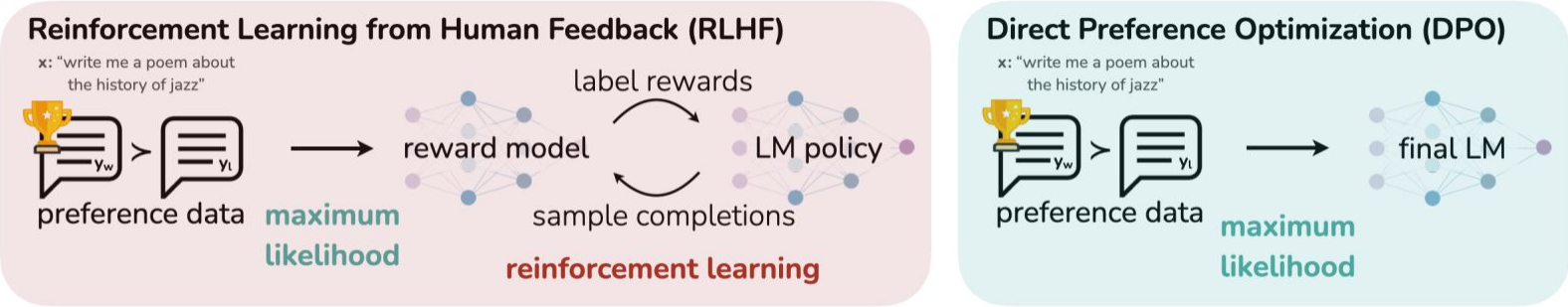
$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$



Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Direct Preference Optimization

Direct Preference Optimization



Rafailov, Rafael, et al. "Direct preference optimization: Your language model is secretly a reward model." *Advances in Neural Information Processing Systems* 36 (2023): 53728-53741.

Direct Preference Optimization

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right].$$

Ratio de prob. de la respuesta preferida, y_w

Ratio de prob. de la respuesta rechazada, y_l

Rafailov, Rafael, et al. "Direct preference optimization: Your language model is secretly a reward model." *Advances in Neural Information Processing Systems* 36 (2023): 53728-53741.

Direct Preference Optimization

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right].$$

Ratio de prob. de la respuesta preferida, y_w

Ratio de prob. de la respuesta rechazada, y_l

Rafailov, Rafael, et al. "Direct preference optimization: Your language model is secretly a reward model." *Advances in Neural Information Processing Systems* 36 (2023): 53728-53741.

Entrenamiento Eficiente de LLMs

Métodos eficientes en parámetros

Cuantización

Entrenamiento Eficiente de LLMs

- Entrenar un LLM se vuelve restrictivo sin los recursos apropiados
  - **80GB** para entrenar 7B en precision bfloat16

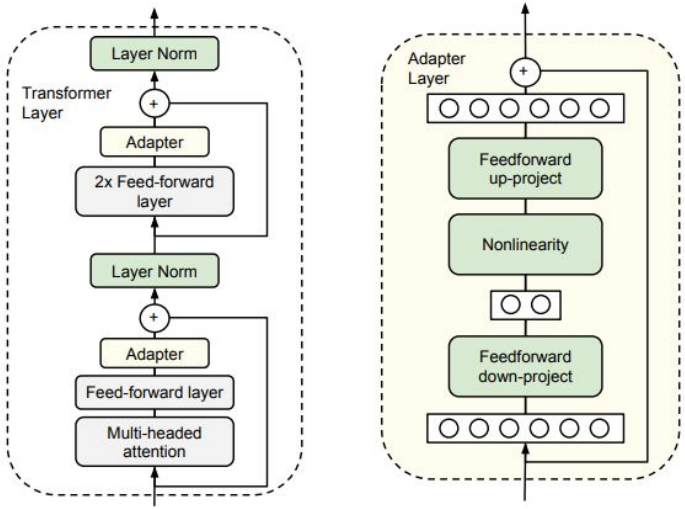
- Soluciones (campo activo de investigacion)
  - Entrenar ***solo una cantidad pequeña*** de parámetros en vez de toda la NN
  - ***Cuantizar*** los parámetros

https://huggingface.co/docs/peft/index

Houlsby, Neil, et al. "Parameter-efficient transfer learning for NLP." *International conference on machine learning*. PMLR, 2019.

Adapters (Houlsby et al. 2019)

- Introduce un “module adaptativo” de cuello de botella (“*auto-encoder*”)
- **Solo el adapter es entrenado**, el resto del LLM se congela



Houlsby, Neil, et al. "Parameter-efficient transfer learning for NLP." *International conference on machine learning*. PMLR, 2019.

Adapters (Houlsby et al. 2019)

- Entrenando solo los adapters mantiene el 95% del performance comparado a entrenar toda las red
- Evaluado con BERT y tareas lingüísticas, semanticas

	Total num params	Trained params / task	CoLA	SST	MRPC	STS-B	QQP	MNLI_m	MNLI_mm	QNLI	RTE	Total
BERT_LARGE	9.0×	100%	60.5	94.9	89.3	87.6	72.1	86.7	85.9	91.1	70.1	80.4
Adapters (8-256)	1.3×	3.6%	59.5	94.0	89.5	86.9	71.8	84.9	85.1	90.7	71.5	80.0
Adapters (64)	1.2×	2.1%	56.9	94.2	89.6	87.3	71.8	85.3	84.6	91.4	68.8	79.6

Houlsby, Neil, et al. "Parameter-efficient transfer learning for NLP." *International conference on machine learning*. PMLR, 2019.

Adapters (Houlsby et al. 2019)

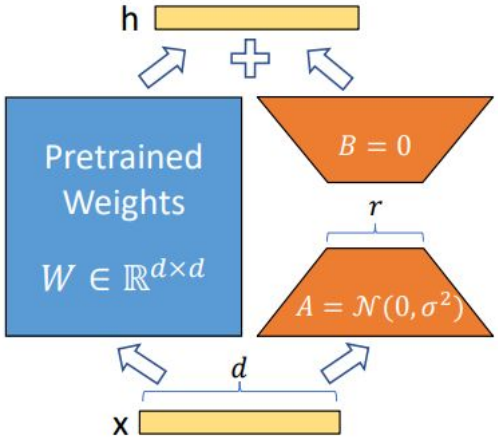
- Adapters puede adaptarse a nuevos lenguajes
  - Entrenando BLOOM en lenguajes no vistos durante pre-entrenamiento

	Models	Strategies	Ckpt.	Emb.	Adpt. Red.	(p.) de	en→ de	de→ de	(p.) ko	en→ ko	ko→ ko
(1)	$mBERT_{BASE}$	-	-	-	-	-	70.0	75.5	-	69.7	72.9
(2)	$XLMR_{LARGE}$	-	-	-	-	-	82.5	85.4	-	80.4	86.4
(3)	$XGLM_{1.7B}$	-	-	-	-	45.4	-	-	45.17	-	-
(4)	BigScience	-	-	-	-	34.1	44.8	67.4	-	-	-
(5)	BigScience	Emb	118,500	wte,wpe	-	41.4	50.7	74.3	34.4	45.6	53.4
(6)	BigScience	Emb→Adpt	118,500	wte,wpe	16	40.0	50.5	69.9	33.8	40.4	51.8
(7)	**BigScience**	**Emb+Adpt**	**118,500**	**wte**	**16**	**42.4**	**58.4**	**73.3**	**38.8**	**49.7**	**55.7**
(8)	BigScience	Emb+Adpt	118,500	wte	48	42.4	57.6	73.7	36.3	48.3	52.9
(9)	BigScience	Emb+Adpt	118,500	wte	384	42.4	55.3	74.2	37.5	49.4	54.6
(10)	BigScience	Emb+Adpt	100,500	wte	16	44.3	56.9	73.2	37.5	48.6	50.8
(11)	BigScience	Emb+Adpt	12,000	wte	16	33.5	55.2	70.5	32.9	46.4	53.3
(12)	BigScience	Emb+Adpt	100,500	wte,wpe	16	-	-	-	37.5	53.5	63.5
(13)	BigScience	Emb+Adpt	118,500	wte,wpe	16	44.7	64.9	73.0	-	-	-

Yong, Zheng Xin, and Vassilina Nikoulina. "Adapting BigScience Multilingual Model to Unseen Languages." *Challenges {\&} Perspectives in Creating Large Language Models*.

LoRa: Low-Rank Adaptation

- Low-rank : matrice A,B tienen rango bajo (r << r(W))
- Agrega adapters en paralelo a las capas lineales
  - Parametros W^Q, W^K, W^V en los módulos de atención
  - Mas no en modulos FF/MLP, embeddings, logits



Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." *ICLR* 1.2 (2022): 3.

LoRa: Low-Rank Adaptation

Model&Method	# Trainable Parameters	WikiSQL Acc. (%)	MNLI-m Acc. (%)	SAMSum R1/R2/RL
GPT-3 (FT)	175,255.8M	**73.8**	89.5	52.0/28.0/44.5
GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5
GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 ($\text{Adapter}^{\text{H}}$)	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 ($\text{Adapter}^{\text{H}}$)	40.1M	73.2	**91.5**	53.2/29.0/45.1
GPT-3 (LoRA)	4.7M	73.4	**91.7**	**53.8/29.8/45.9**
GPT-3 (LoRA)	37.7M	**74.0**	**91.6**	53.4/29.2/45.1

Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." *ICLR* 1.2 (2022): 3.

Cuantización de parámetros

Problema

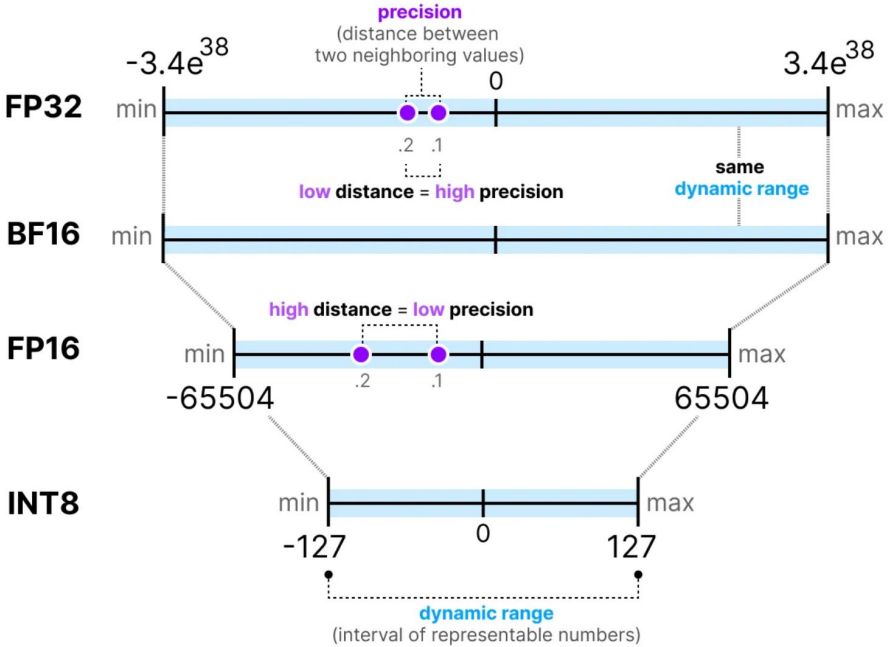
- Comunmente, los pesos de un modelo son representados por floats de 32, 16 bits
  - En pytorch: float32, float16, *bfloat16*
- Cargar todos lo parámetros y gradientes en memoria se vuelve restrictivo
  - El optimizador require ~2x memoria GPU que los parametros!

Solucion

- *Cuantizar* parámetros y representarlos con menos #bits
  - Int8 (8 bits), int4 (4 bits)

Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." *ICLR* 1.2 (2022): 3.

Cuantización de parámetros



Memoria = precision bits * # parametros / 8

Para un modelo con 70B params

FP64 -> 560GB
FP32 -> 280GB (*full precision*)
FP16 -> 140GB

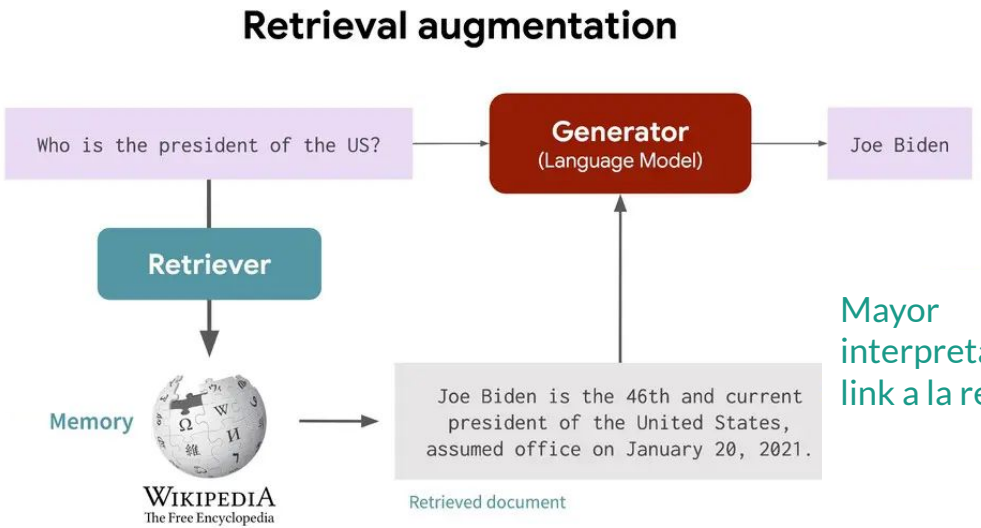
Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models." *ICLR* 1.2 (2022): 3.

Interfaces de Inferencia, RAG

Interfaces

Retrieval-Augmented Generation

Base de datos puede ser actualizada o reemplazada



Mayor interpretabilidad: link a la referencia

Tarea 2

Cuestionario final

https://docs.google.com/forms/d/e/1FAIpQLScyU4N9WgBG1LA4j0Tt_gBTXl8yWzR_xzAQ2vsRfpE77p5PXw/viewform?usp=sharing

Preguntas?