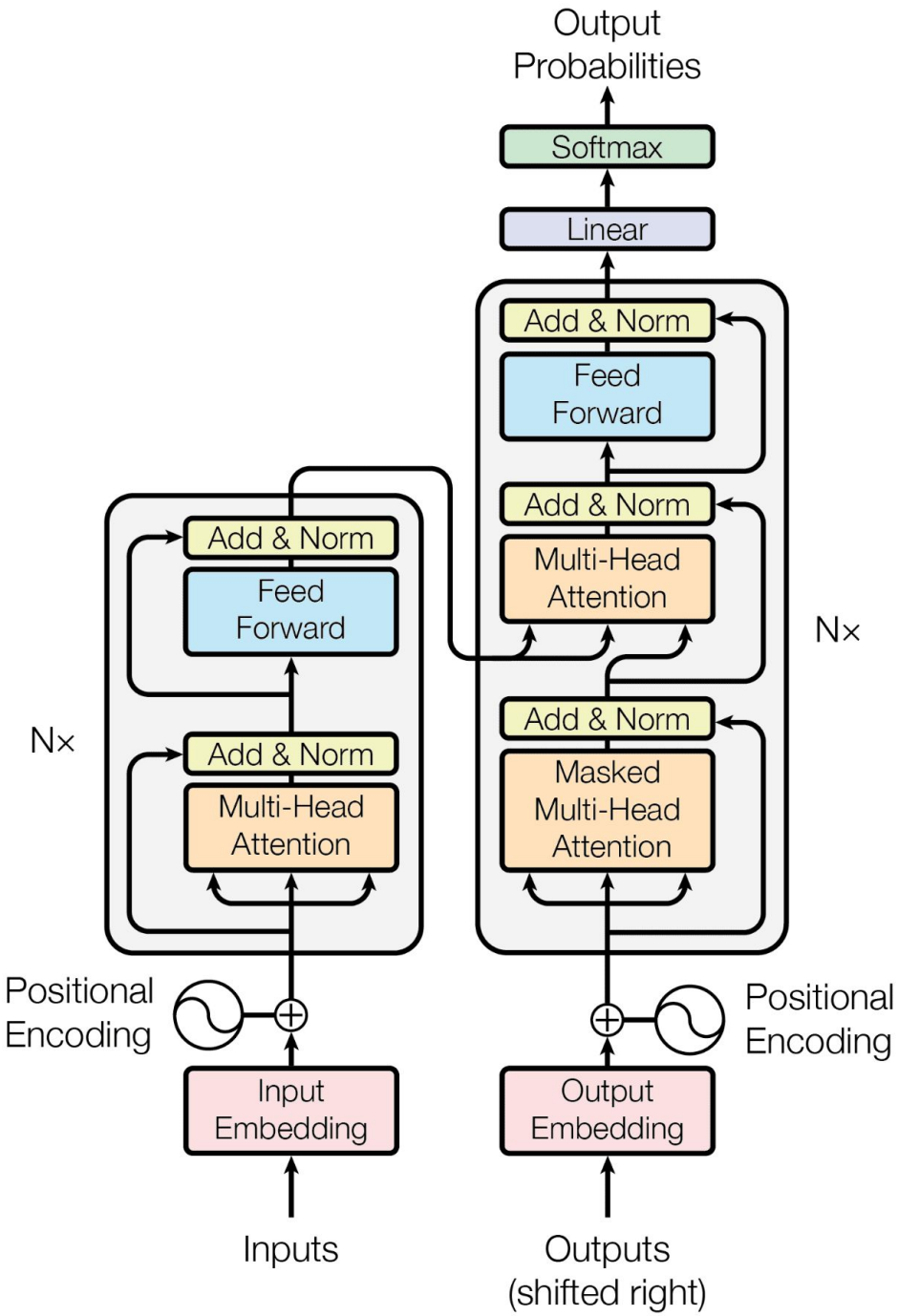
Redes Neuronales para Lenguaje Natural

2024

Grupo de Procesamiento de Lenguaje Natural
Instituto de Computación


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Formas de uso

El transformer original fue planteado como arquitectura encoder-decoder

Primer ejemplo de uso: traducción automática

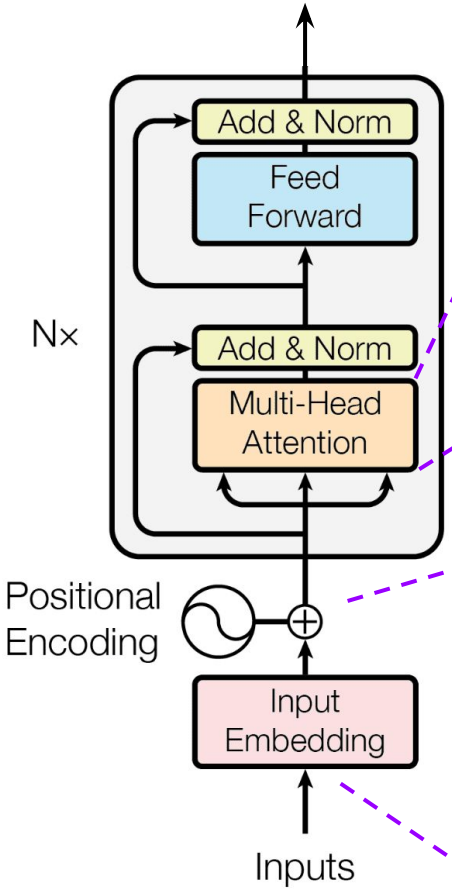
Pero luego aparecen otras variantes

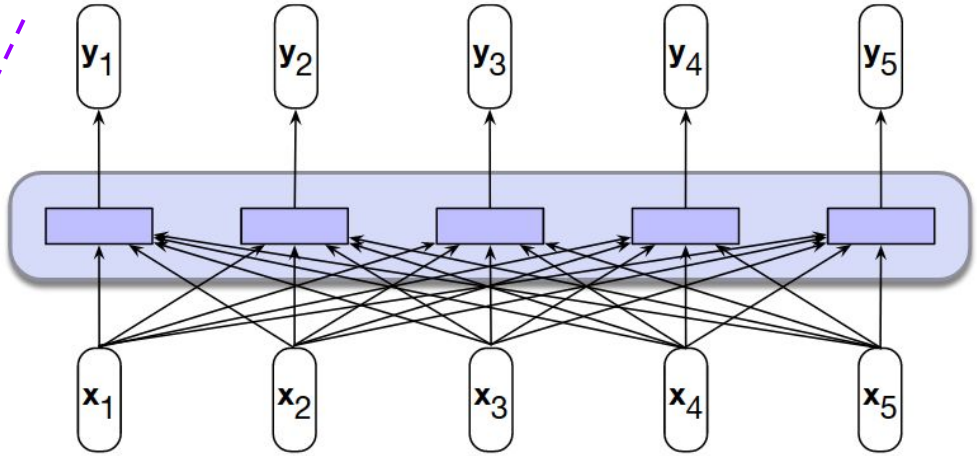
Encoder-only (por ejemplo BERT)

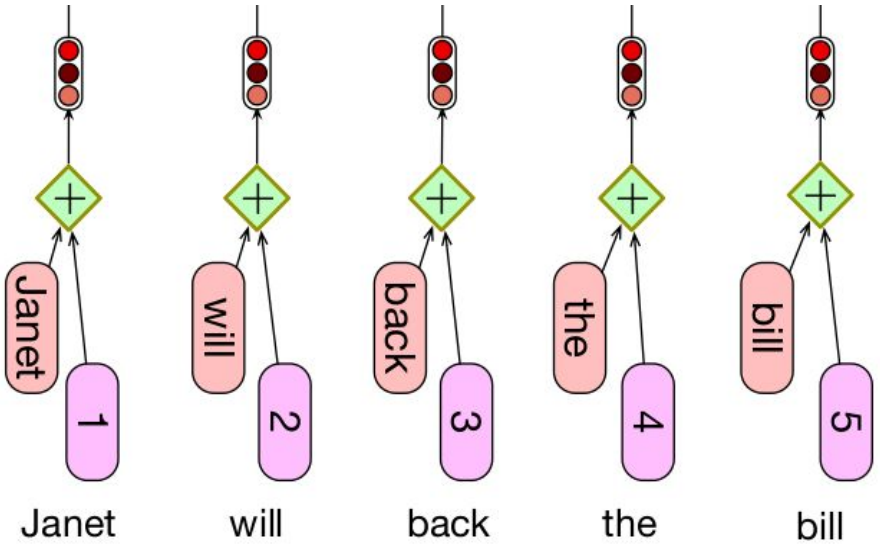
Decoder-only (por ejemplo GPT)

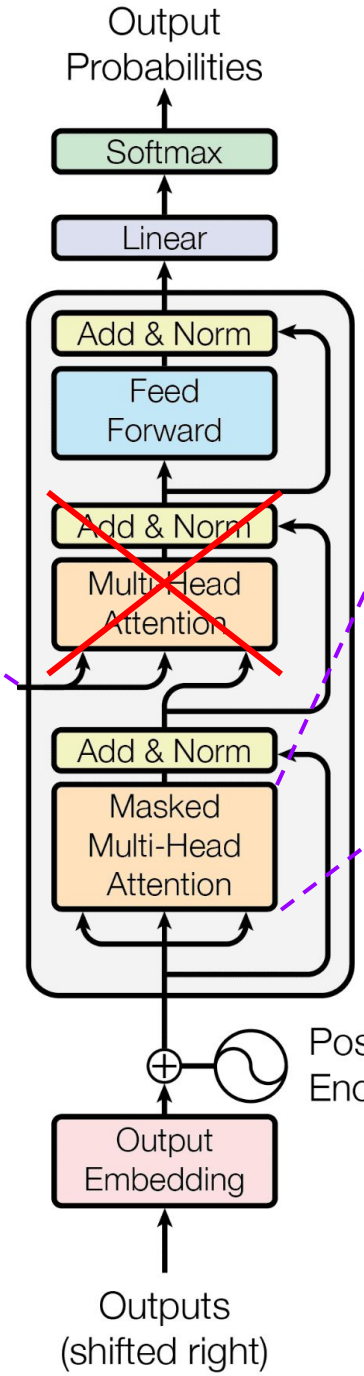
GPT: Modelos Decoder-only

…
Varias capas, pero siempre del mismo tamaño de salida
…



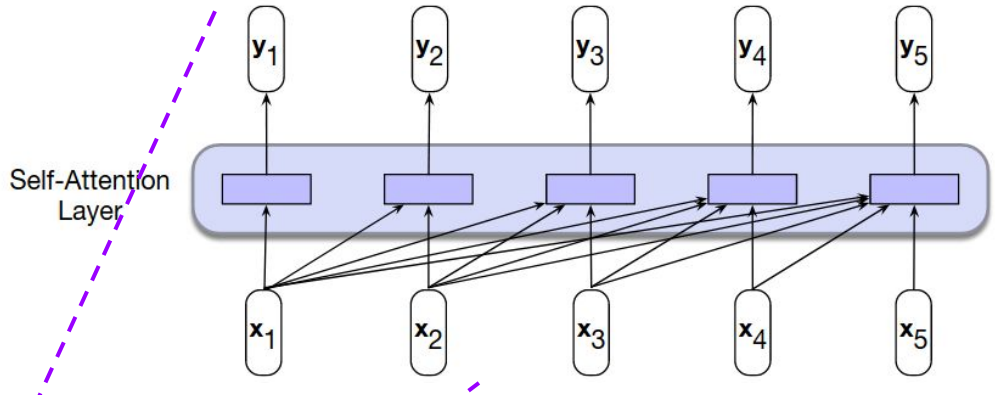






Como no hay encoder, este paso se saltea

N×



$$\mathbf{Q} = \mathbf{X}\mathbf{W}^{\mathbf{Q}};\ \mathbf{K} = \mathbf{X}\mathbf{W}^{\mathbf{K}};\ \mathbf{V} = \mathbf{X}\mathbf{W}^{\mathbf{V}}$$

$$\mathit{SelfAttention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{softmax}\left(\mathrm{mask}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathsf{T}}}{\sqrt{d_k}}\right)\right)\mathbf{V}$$

N

q1·k1	−∞	−∞	−∞
q2·k1	q2·k2	−∞	−∞
q3·k1	q3·k2	q3·k3	−∞
q4·k1	q4·k2	q4·k3	q4·k4

N

Modelos de lenguaje

Hemos visto varios acercamientos a los modelos de lenguaje en el curso

- N-gramas, ventana de palabras con MLP, redes recurrentes, y ahora transformer

Siempre la idea es obtener la probabilidad de una frase o (análogamente) predecir la siguiente palabra

Los transformers tipo decoder-only se usan principalmente para (grandes) modelos de lenguaje (**LLM**)

Modelos de lenguaje

A pesar de que están preentrenados para predecir la siguiente palabra, aprenden un montón de conocimiento sobre el idioma

Porque se entrenan con **muchísimo** texto

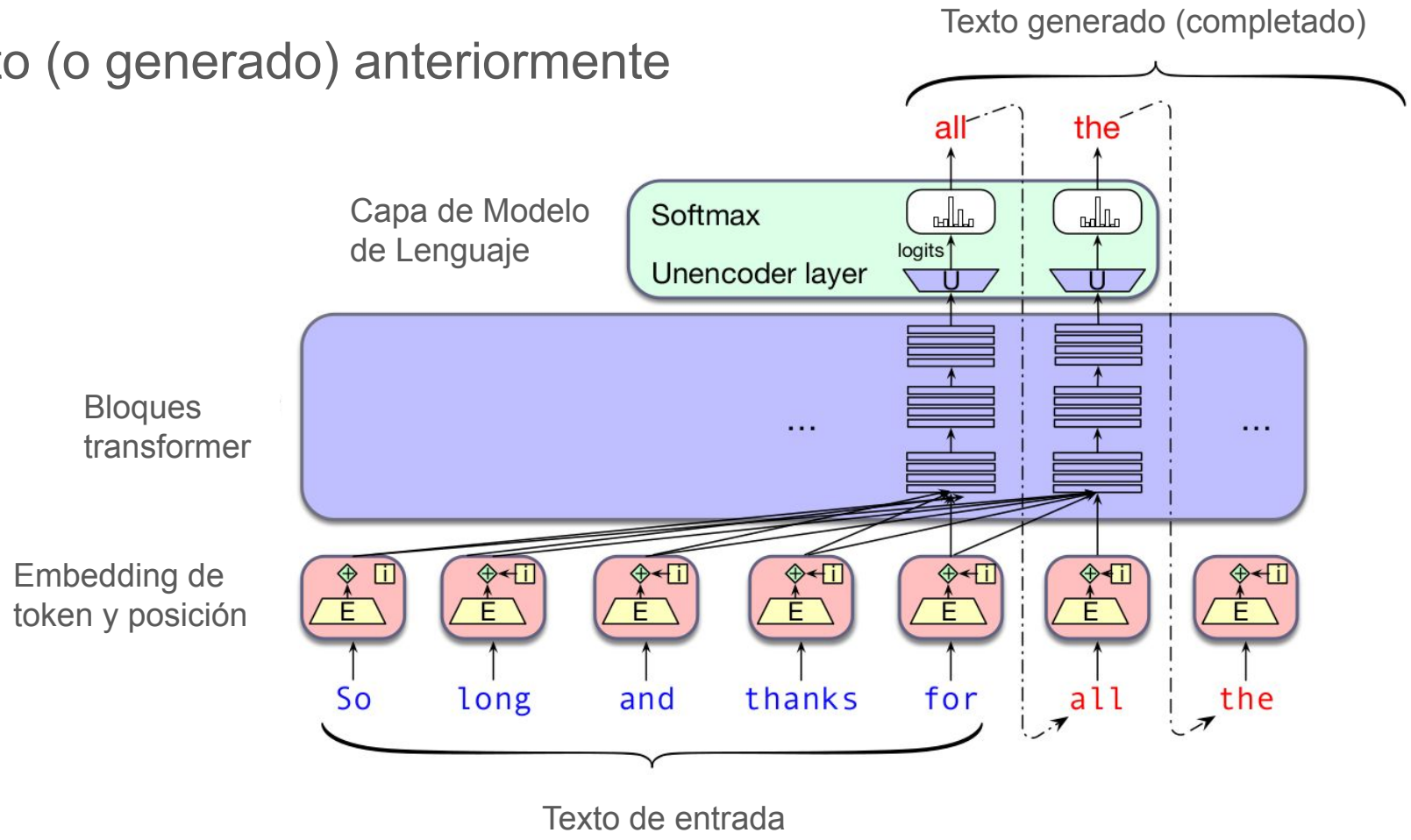
> La arquitectura altamente paralelizable permite que se entrenen con más parámetros, con más texto, y durante más tiempo y seguir obteniendo mejoras en la loss

Muchas tareas de PLN se pueden transformar en tareas de predicción de palabras!

¿Cómo?

Generación condicional

Generar un texto condicionado al texto visto (o generado) anteriormente



Tareas

Muchas tareas de PLN se pueden reformular como un problema de predicción de palabras

Por ejemplo:

- Análisis de sentimiento
- Respuestas a preguntas
- Resúmenes automáticos

Tareas - Análisis de Sentimiento

Análisis de sentimiento: “Me gusta Jackie Chan”

1. Le damos al modelo de lenguaje el string:

   El sentimiento de la oración “Me gusta Jackie Chan” es:

2. Y miramos la distribución de las palabras que vienen luego:

   P(positivo | El sentimiento de la oración “Me gusta Jackie Chan” es:)

   P(negativo | El sentimiento de la oración “Me gusta Jackie Chan” es:)

Tareas - QA

Respuestas a Preguntas (QA): “Quién escribió El Origen de las Especies”

1. Le damos al modelo de lenguaje el string:

   P: ¿Quién escribió el libro “El Origen de las Especies”? R:

2. Y miramos la las palabras que cree que vienen luego:

   P(w| P: ¿Quién escribió el libro “El Origen de las Especies”? R:)

3. 3. Iteramos

   P(w| P: ¿Quién escribió el libro “El Origen de las Especies”? R: Charles)

Tareas - Resumen Automático

Resumen automático usando *tl;dr*

En varias plataformas web que permiten enviar textos largos, los usuarios usan la notación informal “tl;dr” (*too long, didn’t read*) para luego escribir un resumen de lo que escribieron

Los LLMs han visto muchos ejemplos de uso de este token, por lo que podemos usarlo para generar un resumen

Tareas - Resumen Automático

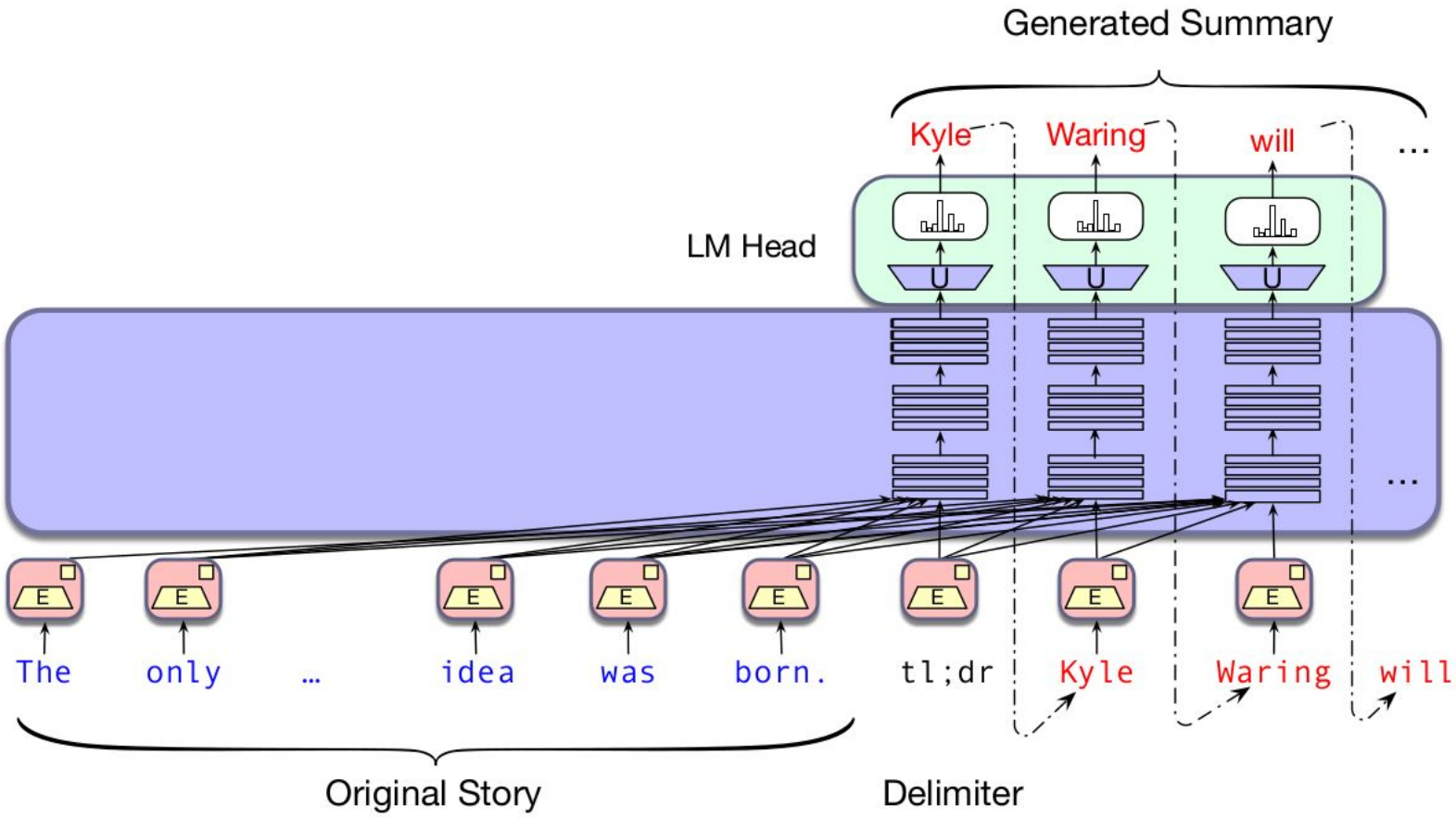
El Instituto Uruguayo de Meteorología, Inumet, dejó sin efecto las alertas meteorológicas que emitiera a primera hora de la jornada. La institución había advertido acerca de una perturbación atmosférica asociada a masa de aire húmeda e inestable, capaz de generar tormentas, algunas de ellas "puntualmente fuertes". Las advertencias se enmarcaban en lo previsto por el aviso especial que la entidad emitiera en la tarde del pasado domingo. En su nueva actualización, Inumet informa que las condiciones atmosféricas han mejorado en forma temporaria, por lo que se determina el cese de la advertencia vigente. Sin embargo, "se mantiene el monitoreo de la situación y en caso de ser necesario se emitirán nuevas advertencias meteorológicas", detalla la entidad.

tl;dr

Inumet canceló las alertas meteorológicas emitidas anteriormente debido a una mejora temporal en las condiciones atmosféricas. Aunque no hay advertencias vigentes, se sigue monitoreando la situación y se emitirán nuevas alertas si es necesario.

Noticia de Montevideo Portal del 15/10/2024, tl;dr de ChatGPT.

Tareas - Resumen Automático




Decodificación

Decodificación y Muestreo

La tarea de elegir una palabra a generar basada en las probabilidades del modelo se conoce como **decodificación** (*decoding*)

Podríamos siempre obtener la siguiente palabra más probable: estrategia *greedy*

O podríamos usar una estrategia de *beam search* como vimos para traducción automática

Pero en los LLMs se suele utilizar técnicas que introduzcan aleatoriedad y diversidad en las respuestas

Decodificación y Muestreo

Método más común para hacer decoding en LLMs:

muestreo (*sampling*)

Sampling de la distribución de palabras devuelta por un modelo:

- Elegir una palabra aleatoriamente de acuerdo a las probabilidades asignadas por el modelo
- Luego de generar cada token, hacemos sampling del siguiente token de acuerdo a la probabilidad condicionada a las elecciones previas

tomar una muestra aleatoria (*sample*) de acuerdo a la distribución p(w)

$$\mathrm{i} \leftarrow 1$$
$$w_i \sim \mathrm{p}(w)$$
while w_i != EOS
$$\mathrm{i} \leftarrow \mathrm{i} + 1$$
$$w_i \sim \mathrm{p}(w_i \mid w_{<i})$$

Sampling aleatorio

Aunque el sampling aleatorio suele generar con alta probabilidad palabras coherentes con el contexto

Hay muchísimas palabras más raras con baja probabilidad en la cola de la distribución

Cada una tiene muy baja probabilidad, pero al sumarlas son una gran parte de la distribución

Lo suficiente como para que con mucha frecuencia se generen oraciones sin sentido o muy extrañas

Sampling

Factores: calidad vs. diversidad

Si enfatizamos palabras de alta probabilidad

+ calidad: más exacto, coherente, fáctico

- diversidad: aburrido, repetitivo

Si enfatizamos las palabras de probabilidad media

+ diversidad: más creativo, textos más variados y diferentes

- calidad: menos fáctico, incoherente, propenso a inventar

Sampling Top-k

1. Elegir una cantidad de palabras k

2. Para cada palabra del vocabulario V, usar el modelo de lenguaje para predecir la probabilidad de esta palabra dado el contexto $p(w_i | w_{<i})$

3. Ordenar palabras por probabilidad, y quedarse solo con las k más probables

4. Renormalizar los puntajes de las k más probables (para que sea distribución de probabilidad)

5. Elegir aleatoriamente una palabra de estas k más probables de acuerdo a sus probabilidades

Sampling Top-p (nucleus sampling)

Problema del top-k: hay un k fijo para cubrir distribuciones con diferentes masas de probabilidad en distintas situaciones

Idea: En vez de esto, mantener *"p"* por ciento de la masa de probabilidad

Dada la distribución $P(w_i \mid w_{<i})$, el vocabulario top-p V(p) es el conjunto de palabras más pequeño que cumple

$$\sum_{w \in V^{(p)}} P(w|\mathbf{w}_{<t}) \geq p$$

Sampling con Temperatura

En vez de truncar la distribución, cambia su forma

$$y = \text{softmax}(u) \quad \Rightarrow \quad y = \text{softmax}(u / \tau)$$

El sampling de baja temperatura ($\tau < 1$) causa lo siguiente:

- aumenta la probabilidad de las palabras más probables
- disminuye la probabilidad de las palabras más raras

Cuando τ es cercano a 1, la temperatura no afecta mucho

¿Qué pasa cuando es mucho mayor a 1?

Cache KV

Durante el entrenamiento, podemos paralelizar el cálculo de la atención para que sea más eficiente

$$SelfAttention(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{softmax}\left(\text{mask}\left(\frac{\mathbf{Q}\mathbf{K}^\mathsf{T}}{\sqrt{d_k}}\right)\right)\mathbf{V}$$

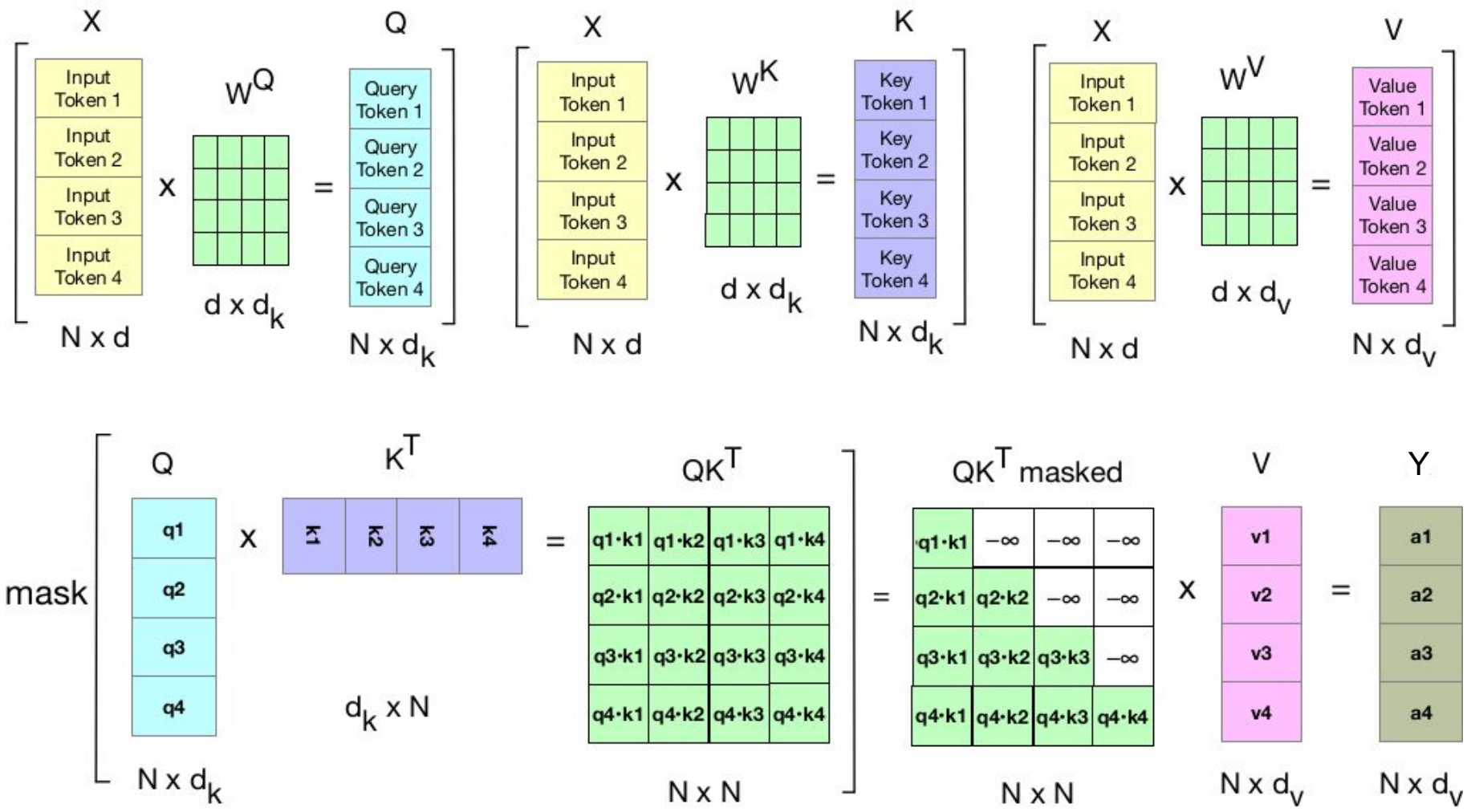
Pero no podemos hacerlo al momento de inferencia, porque generamos un token a la vez

Para cada token nuevo x_i generado, cuando queremos procesar el siguiente tenemos que multiplicarlo por W^Q, W^K y W^V para obtener los vectores *q*, *k*, *v*

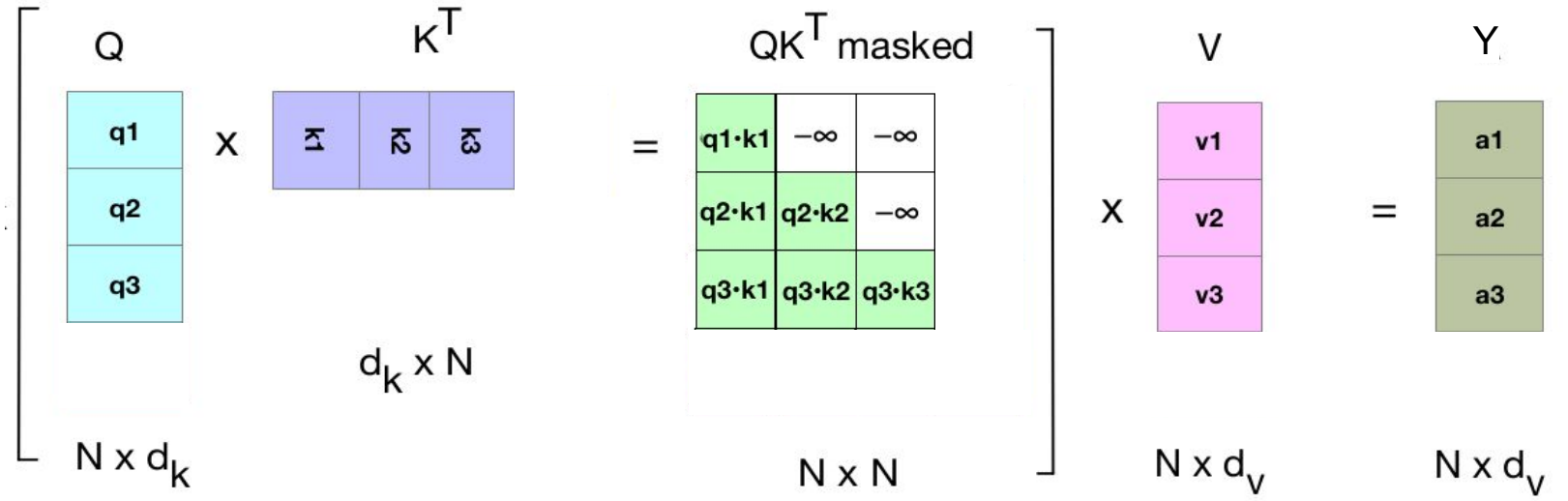
Pero no queremos volver a recalcular los vectores *k* y *v* para los tokens anteriores $x_{<i}$

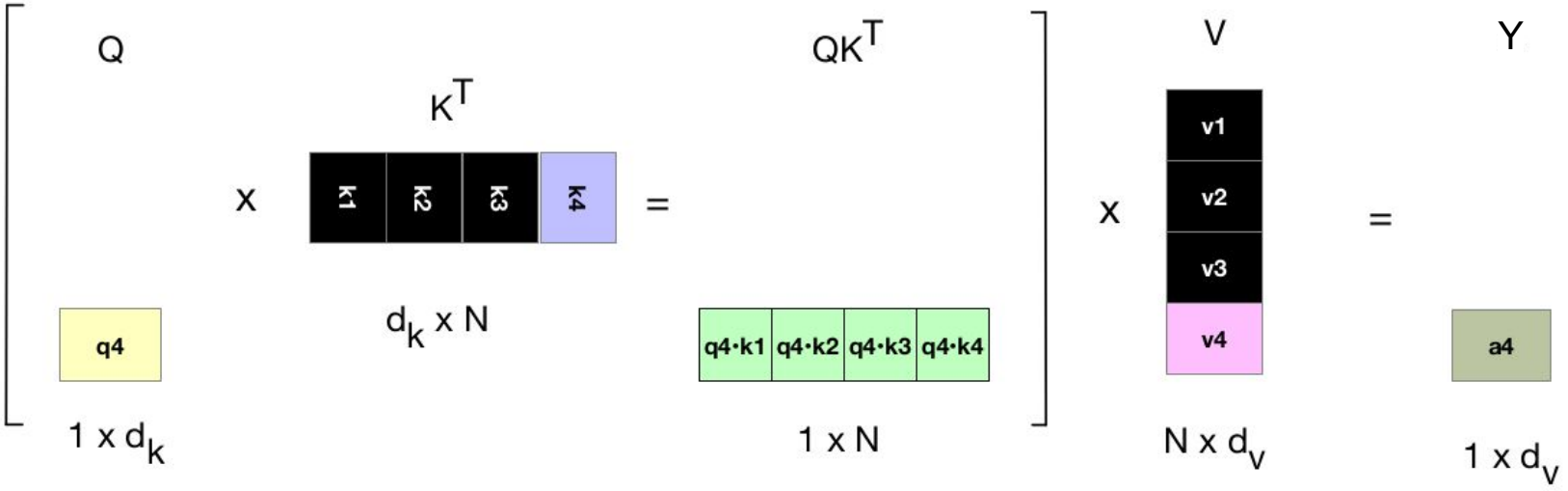
En vez de eso, almacenamos los vectores *k* y *v* en memoria en el llamado **cache KV**, y los obtenemos de ahí la siguiente vez

Cálculo en paralelo



Cache KV


Q
K^T
QK^T masked
V
Y
q1
q2
q3
x
k1
k2
k3
=
q1·k1
−∞
−∞
q2·k1
q2·k2
−∞
q3·k1
q3·k2
q3·k3
x
v1
v2
v3
=
a1
a2
a3
d_k x N
N x d_k
N x N
N x d_v
N x d_v


Q
K^T
QK^T
V
Y
x
k1
k2
k3
k4
=
x
v1
v2
v3
v4
=
q4
q4·k1
q4·k2
q4·k3
q4·k4
a4
d_k x N
1 x d_k
1 x N
N x d_v
1 x d_v

Tipos de problemas

1 palabra (o un par) → 1 categoría

frío y *caliente* son sinónimos?
antónimos?
MLP

n palabras → 1 categoría (k posibles)

este tweet tiene sentimiento positivo?
este tweet es un chiste?
MLP, CNN, RNN, BERT

n palabras → 0..k categorías

de qué temas habla este texto?
qué emociones presenta este tweet?
MLP, CNN, RNN, BERT

n palabras → n categorías

POS-tagging, NER,
chunking, parsing
CNN, RNN, BERT

n palabras → m palabras

traducción automática
respuestas a preguntas
resúmenes automáticos
Encoder-Decoder (RNN, Transformer)
Decoder-only (Transformer)


GPT

GPT

Generative Pretrained Transformer

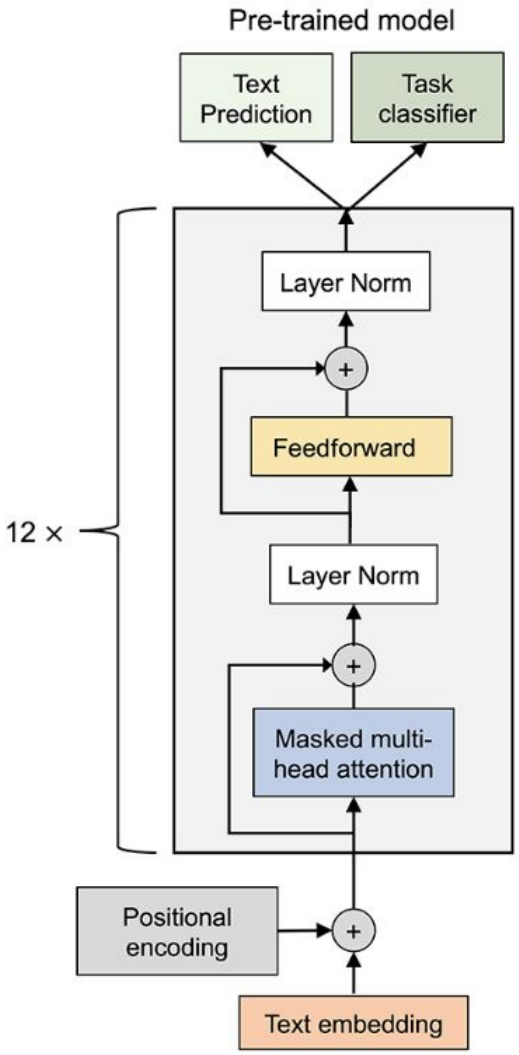
Es una familia de modelos decoder-only muy utilizada

GPT-1 es de 2018, con 110M parámetros

Preentrenado con un gran corpus de texto

Pero luego fine-tuneado para varias tareas:

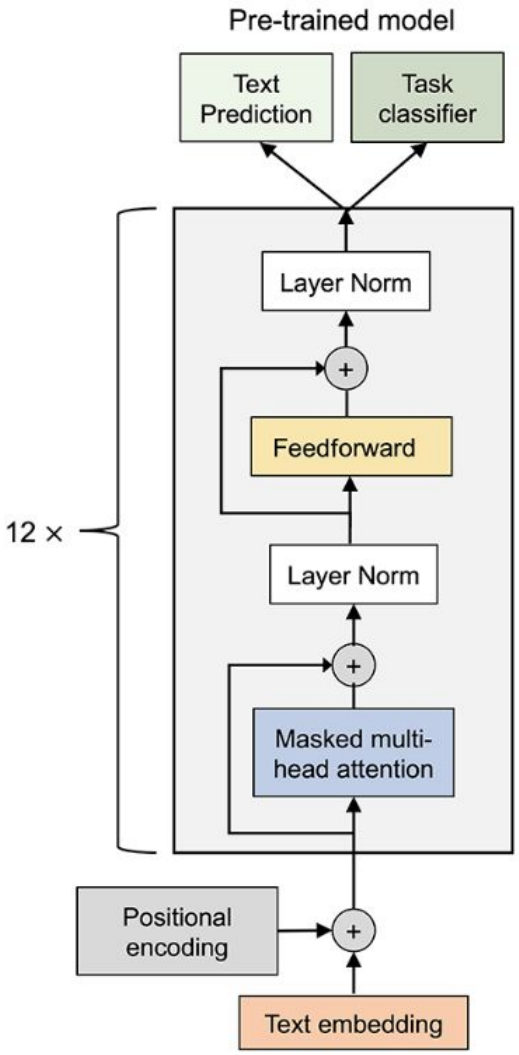
- Clasificación, inferencia textual, similitud, multiple opción



Improving Language Understanding by Generative Pre-Training
Radford et al., (2018)

GPT-1

- 12 bloques transformer (dimensión de salida 768)
- 12 cabezales de self-attention
- Feed forward de 3072
- Activación GELU (Gaussian Error Linear Unit)
- Tokenización BPE
- Embeddings posicionales entrenables

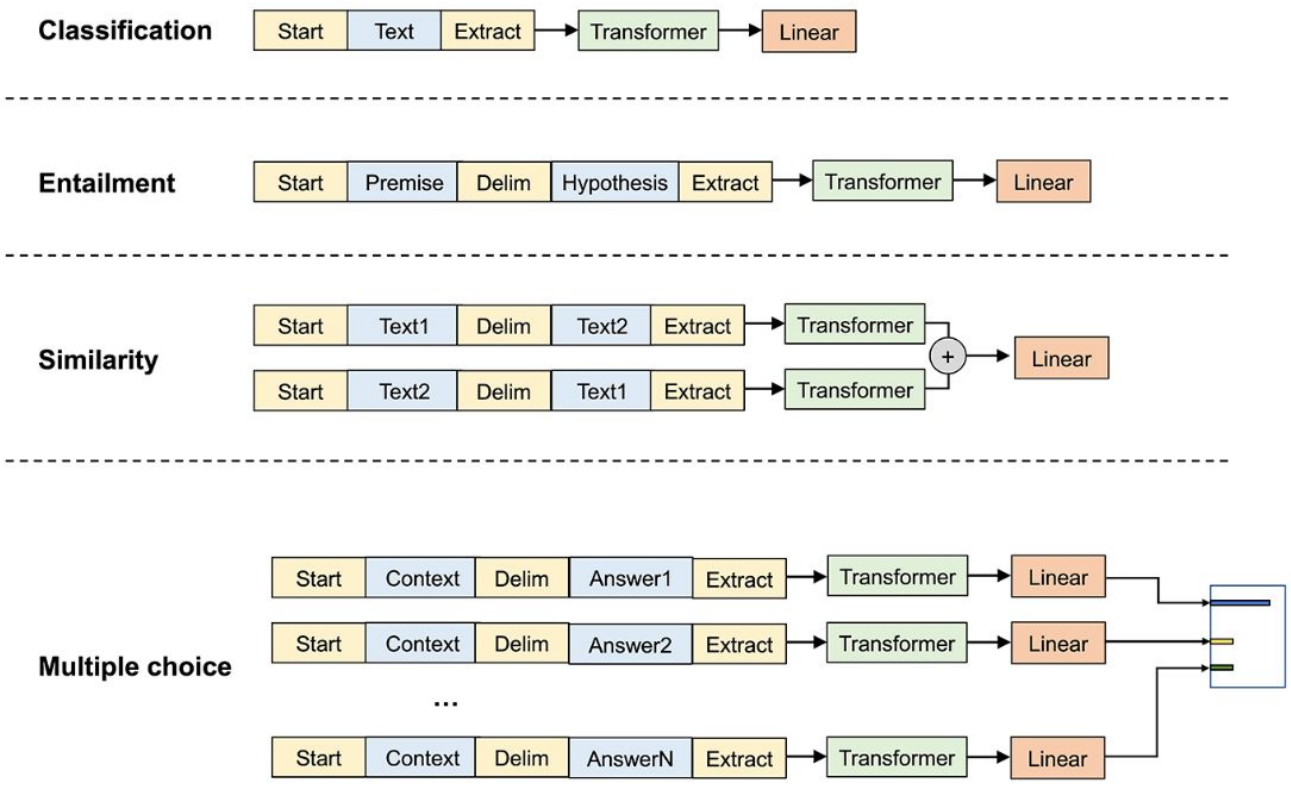


Improving Language Understanding by Generative Pre-Training
Radford et al., (2018)

GPT-1

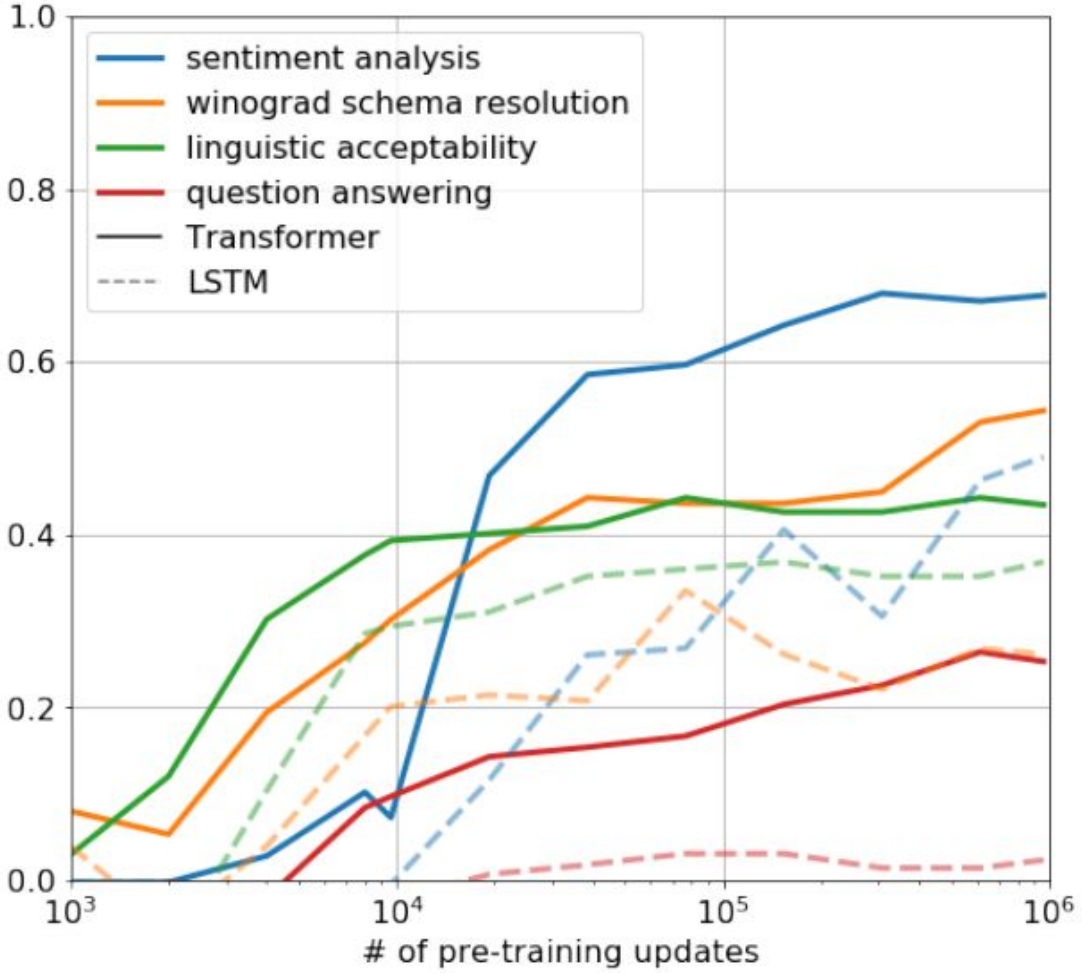
En GPT-1 se hacía una etapa de fine-tuning supervisado

Se usaban capas lineales específicas para adaptarlo a las distintas tareas



GPT-1

Pero ya presentaba capacidades “zero-shot”: Pedir que resuelva una tarea (como responder una pregunta) sin haber visto ningún ejemplo de entrenamiento

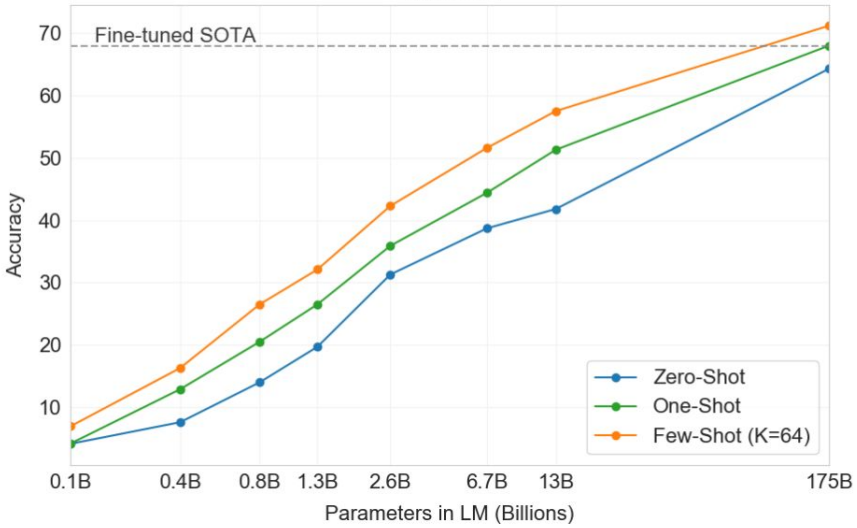


GPT: Evolución

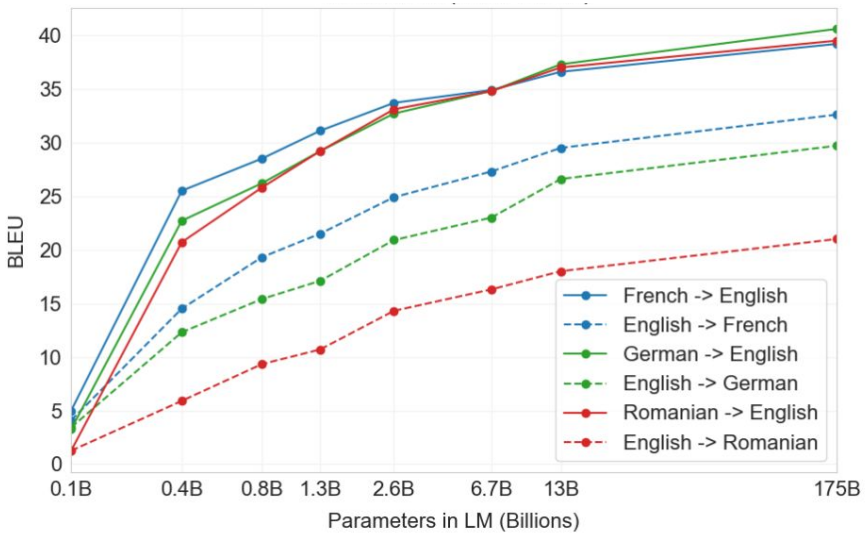
	Año	Parámetros	Capas	Contexto Máximo	Tamaño de embedding
GPT-1	2018	117 M	12	512	768
GPT-2	2019	1.5 B	48	1024	1600
GPT-3	2020	175 B	96	2048	12288
GPT-3.5	2022	175 B	96	2048	12288
GPT-4	2023	1.8 T (?)	120 (?)	8000 (?)	16384 (?)

GPT-3

QA



Traducción Automática



	SuperGLUE Average	BoolQ Accuracy	CB Accuracy	CB F1	COPA Accuracy	RTE Accuracy
Fine-tuned SOTA	**89.0**	**91.0**	**96.9**	**93.9**	**94.8**	**92.5**
Fine-tuned BERT-Large	69.0	77.4	83.6	75.7	70.6	71.7
GPT-3 Few-Shot	71.8	76.4	75.6	52.0	92.0	69.0

	WiC Accuracy	WSC Accuracy	MultiRC Accuracy	MultiRC F1a	ReCoRD Accuracy	ReCoRD F1
Fine-tuned SOTA	**76.1**	**93.8**	**62.3**	**88.2**	**92.5**	**93.3**
Fine-tuned BERT-Large	69.6	64.6	24.1	70.0	71.3	72.0
GPT-3 Few-Shot	49.4	80.1	30.5	75.4	90.2	91.1

Language Models are Few-Shot Learners
Brown et al., (2020)

Otros modelos

La familia de los GPT es muy famosa, pero existen muchos otros modelos decoder-only similares

- Gemini (Google)
- Claude (Anthropic)
- LLaMa (META)
- Mistral (Mistral AI)

  …y otros…

Algunos se pueden descargar, usar, y fine-tunear libremente

Todos vienen en diferentes variantes en cantidad de parámetros (7B, 13B, 70B, 405B,...), cantidad de datos y tiempo de entrenamiento

Cantidad de parámetros

Cómo se calcula la cantidad de parámetros de un modelo?

Normalmente no se cuentan las capas de embeddings (de tokens o posicionales)

Va a depender de la cantidad de capas de bloques transformer, y el tamaño de la capa de atención (tamaño del embedding)

$$N \approx 2 * d * n_{capas} (2 * d_{aten} + d_{ff})$$

$$\approx 12 * n_{capas} * d^2$$

(asumiendo que $d_{aten} = d_{ff} / 4 = d$)

Por ejemplo: GPT-3 tiene N=96 capas y dimensionalidad d=12288 Por lo tanto tiene 12 x 96 x 12288^2 = 175B

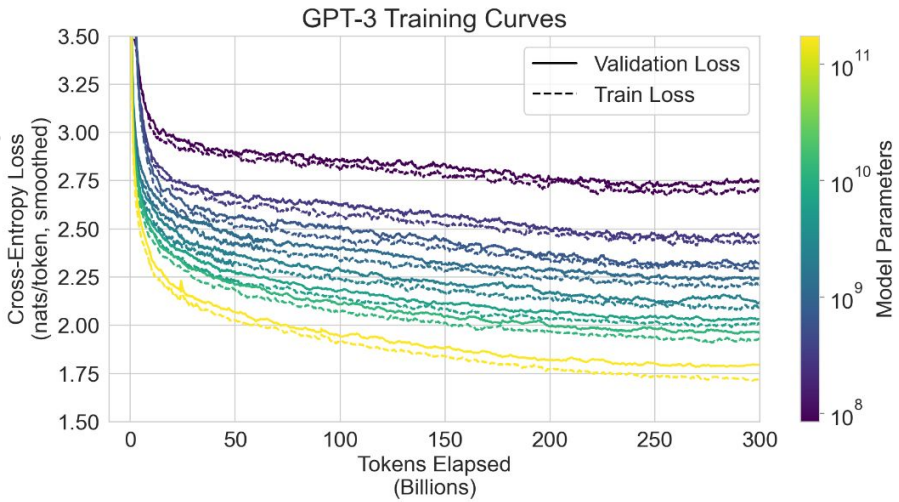
Leyes de escalamiento

La performance de los LLMs depende de:

- Tamaño del modelo: cantidad de parámetros sin contar embeddings
- Tamaño del dataset: cantidad de datos de entrenamiento
- Cantidad de cómputo: uso de unidades y tiempo de cómputo utilizados para entrenar (por ejemplo en FLOPS)

Se puede mejorar un modelo agregando parámetros (más capas, mayor embedding), más datos, o entrenando por más iteraciones

La performance del modelo (medida como loss) escala como una ley de potencia (power-law) con cada uno de estos factores



Leyes de escalamiento

Loss L en función de la cantidad de parámetros N, tamaño del dataset D, cantidad de cómputo C (manteniendo las otras dos constantes)

$$L(N) = \left(\frac{N_c}{N}\right)^{\alpha_N}$$

$$L(D) = \left(\frac{D_c}{D}\right)^{\alpha_D}$$

$$L(C) = \left(\frac{C_c}{C}\right)^{\alpha_C}$$

Power Law	Scale (tokenization-dependent)
$\alpha_N = 0.076$	$N_c = 8.8 \times 10^{13}$ params (non-embed)
$\alpha_D = 0.095$	$D_c = 5.4 \times 10^{13}$ tokens
$\alpha_C = 0.057$	$C_c = 1.6 \times 10^{7}$ PF-days

Se pueden usar las leyes de escalamiento antes de comenzar a entrenar para predecir cuál sería el loss esperado si agregáramos más parámetros o más datos

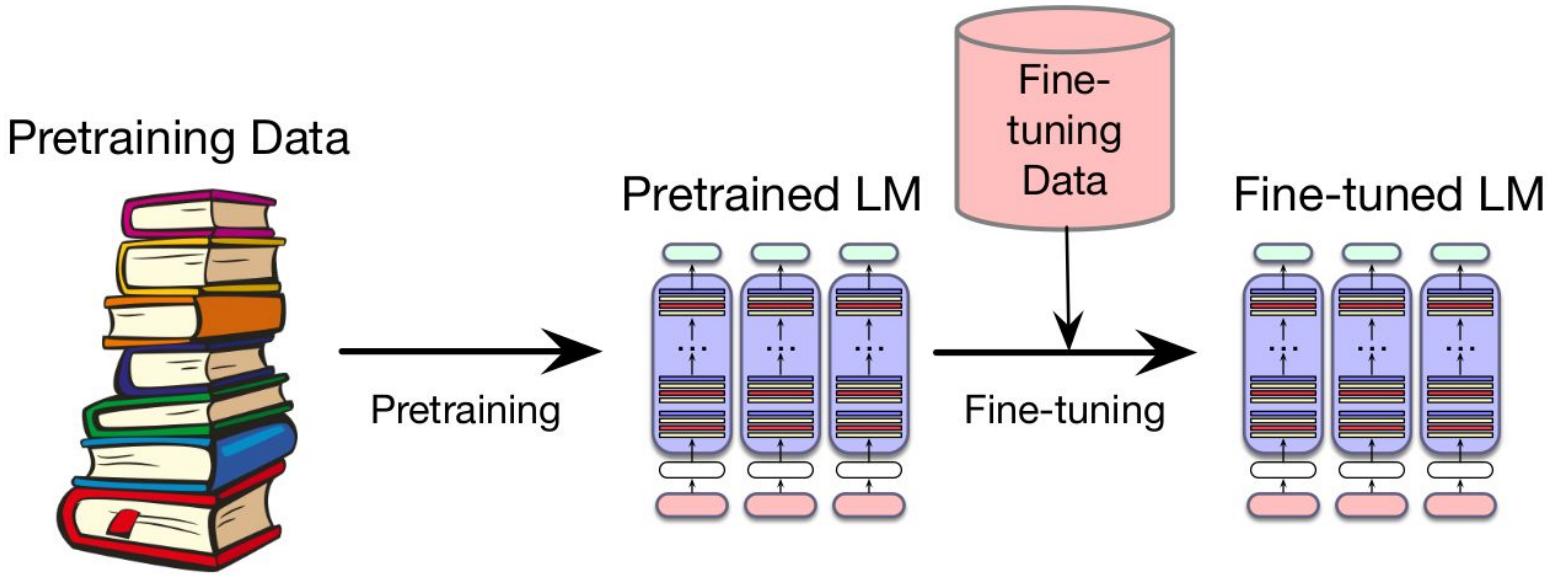
Scaling Laws for Neural Language Models
Kaplan et al., (2020)


Entrenamiento

Entrenamiento: Preentrenamiento y Fine-tuning

Similar a lo que pasaba con BERT (y los modelos encoder-only)

- El entrenamiento de un LLM es un “preentrenamiento”
- Pero ya se puede utilizar tal como está, como vimos antes
- O se puede hacer fine-tuning para adaptarlo mejor a ciertas tareas o idiomas



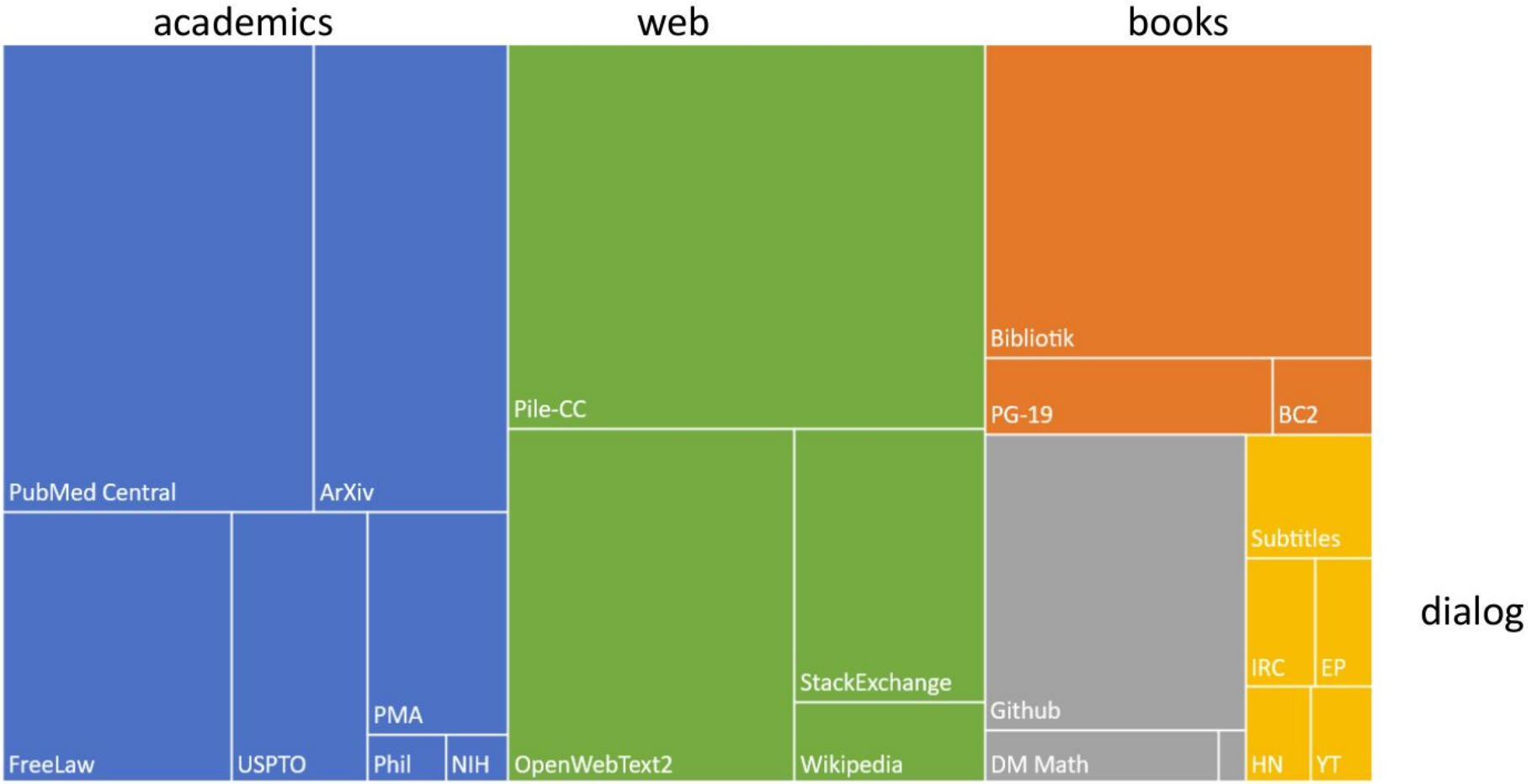
Preentrenamiento

Los LLMs se entrenan con muchísimo texto extraído de la web

- Common crawl, snapshots de la web producidos por la organización Common Crawl, miles de millones de páginas
- Colossal Clean Crawled Corpus (C4; Raffel et al. 2020), 156 mil millones de tokens en English filtrados
- The Pile (Gao et al., 2020), 825 GB de texto en inglés, además de código y artículos académicos

¿Qué contienen? Principalmente documentos, Wikipedia, sitios de noticias, y otras cosas

Preentrenamiento: The Pile



Preentrenamiento: Filtrados

Calidad

- Se intenta clasificar los sitios por calidad, por ejemplo alta calidad son Wikipedia, libros, y algunos sitios particulares confiables
- Eliminación de *boilerplate*: texto repetitivo que aparece en muchas páginas, por ejemplo de relleno o con información sobre el sitio
- Deduplicación en varios niveles (URLs, documentos, hasta líneas)

Seguridad

- Es importante la detección de toxicidad, pero los resultados no han sido del todo buenos
- Se ha visto que variedades de idiomas minoritarias se ven altamente afectadas (ejemplo AAVE)

Preentrenamiento

Qué aprende realmente el modelo del preentreenamiento?

Un montón de cosas!

> *There are canines everywhere! One dog in the front room, and two dogs*
>
> *It wasn't just big it was enormous*
>
> *The author of "A Room of One's Own" is Virginia Woolf*
>
> *The doctor told me that he*
>
> *The square root of 4 is 2*

El texto está lleno de información sobre el funcionamiento de la lengua en sí, también contiene conocimiento específico, y también sesgos

Preentrenamiento: Problemas

Copyright

- Gran cantidad del texto de los datasets tiene copyright
- No es claro si su uso está amparado por leyes de uso justo (*fair use*)
- Es un tema legal no saldado

Consentimiento de los datos

- Los dueños de los sitios pueden indicar que no quieren que se obtengan sus datos
- Se están respetando estas restricciones? Es retroactivo?

Privacidad

- Muchos sitios contienen información privada (mails, IP, cédulas, números de teléfono)
- Se intenta filtrar, pero a veces pueden quedar igual en el dataset

Fine-tuning

Se utiliza el término “fine-tuning” para decir cuatro cosas diferentes

1. Seguir entrenando todo el modelo (continued pretraining) con otros datos
2. Congelar los pesos y solo entrenar unos pocos parámetros: *parameter efficient fine-tuning* (PEFT)
3. Poner una capa extra y solo ajustar esos parámetros para adaptar a otra tarea (como hacíamos en BERT)
4. *Supervised fine-tuning* (SFT) o *Instruction fine-tuning*: adaptación particular usada en ChatGPT y similares, adaptándolo a múltiples tareas

En todos los casos, la idea es: tomar un modelo preentrenado y continuar su entrenamiento adaptando todos o parte de sus parámetros a nuevos datos

Fine-tuning como adaptación a dominio nuevo

Qué hacemos si precisamos que nuestro LLM funcione bien en un dominio que nunca vio durante el preentrenamiento?

- Dominio específico como medicina o derecho
- O un modelo multilingüe que necesita ver más datos en un idioma que aparecía poco durante el preentrenamiento
- Por defecto al tener pocos datos va a tener baja performance

Fine-tuning como adaptación a dominio nuevo

Se siguen entrenando todos los parámetros del modelo con los nuevos datos

- usando el mismo método (predicción de la siguiente palabra) y función de loss (cross-entropy loss) que en el preentrenamiento
- como si los datos nuevos estuvieran al final del corpus de preentrenamiento
- por eso se conoce este tipo de fine-tuning como “continuación del preentrenamiento”

Bibliografía

- Jurafsky and Martin 3rd edition. Capítulo 10. https://web.stanford.edu/~jurafsky/slp3/

- Machine Learning with PyTorch and Scikit-Learn. Raschka et al. 2022. Cap 16.

- Papers...