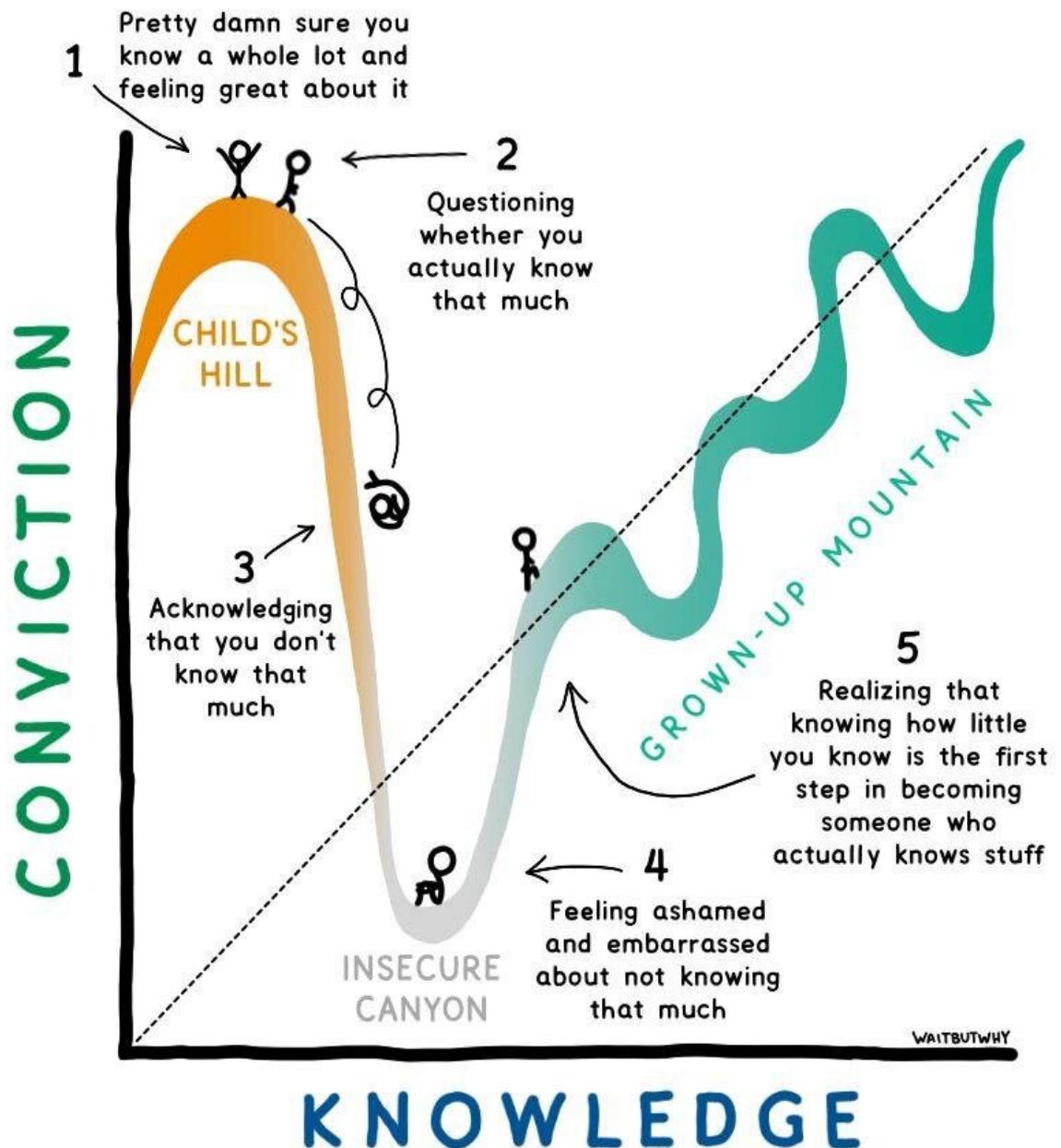


Aprendiendo sobre ~~la Inteligencia Artificial (IA)~~ los Large Language Models (LLMs) a través de los mitos

Universidad Adolfo Ibáñez

Sebastián Moreno
sebastian.moreno@uai.cl

The Dunning-Kruger effect



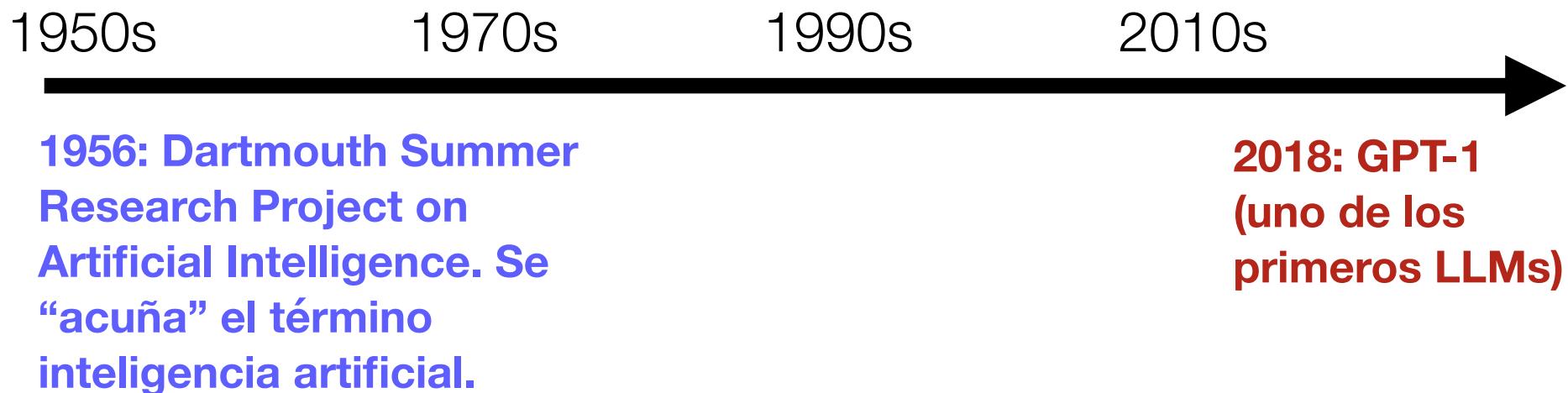
Mito 1: Inteligencia Artificial es lo mismo que Large Language Models

FALSO.

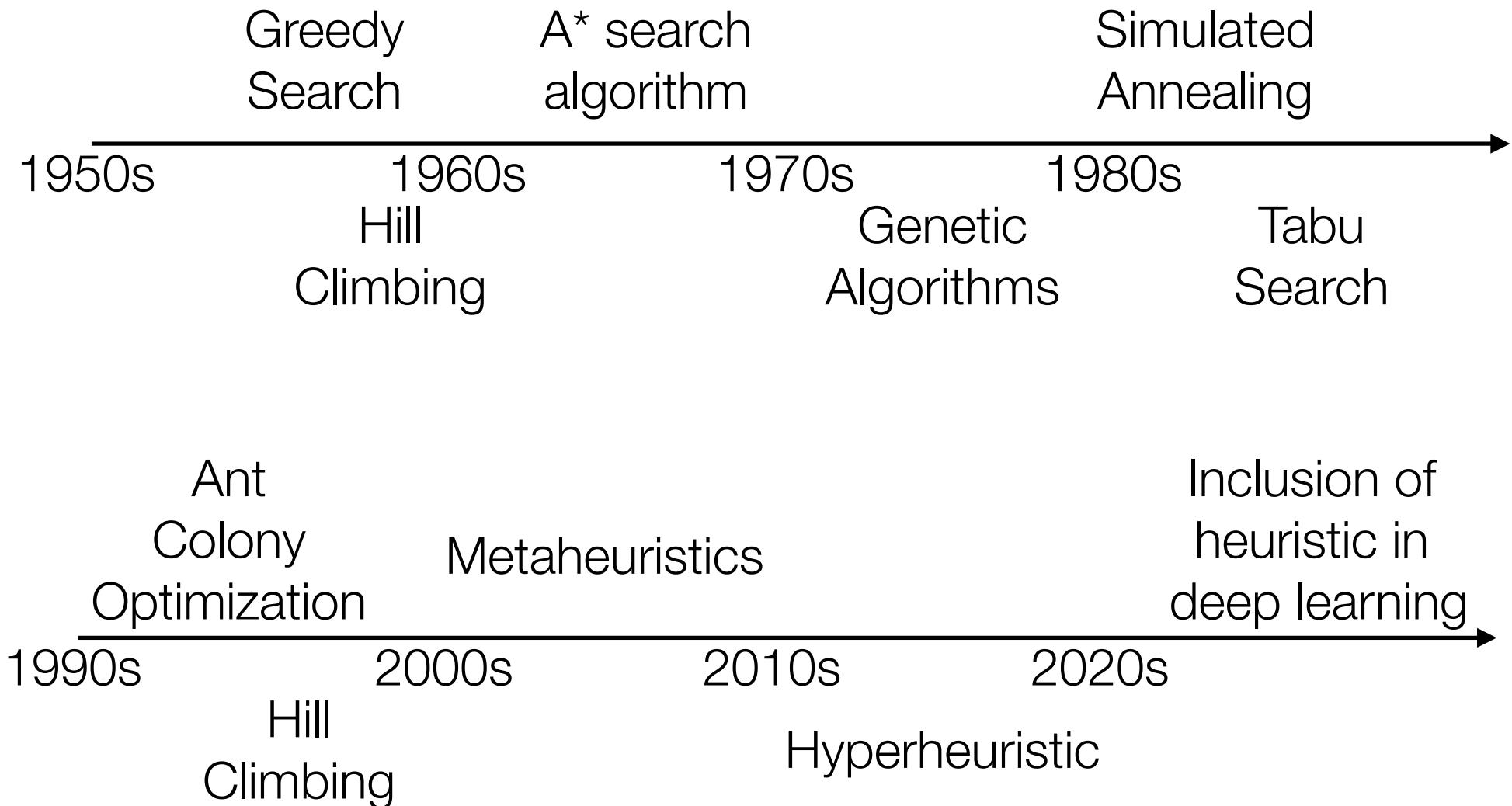
Inicios de la Inteligencia Artificial (IA)

- Antes de 1956: Se menciona el término Inteligencia Artificial y sus posibles orígenes con Alan Turing, McCulloch, and Pitts (diseño de una neurona artificial).

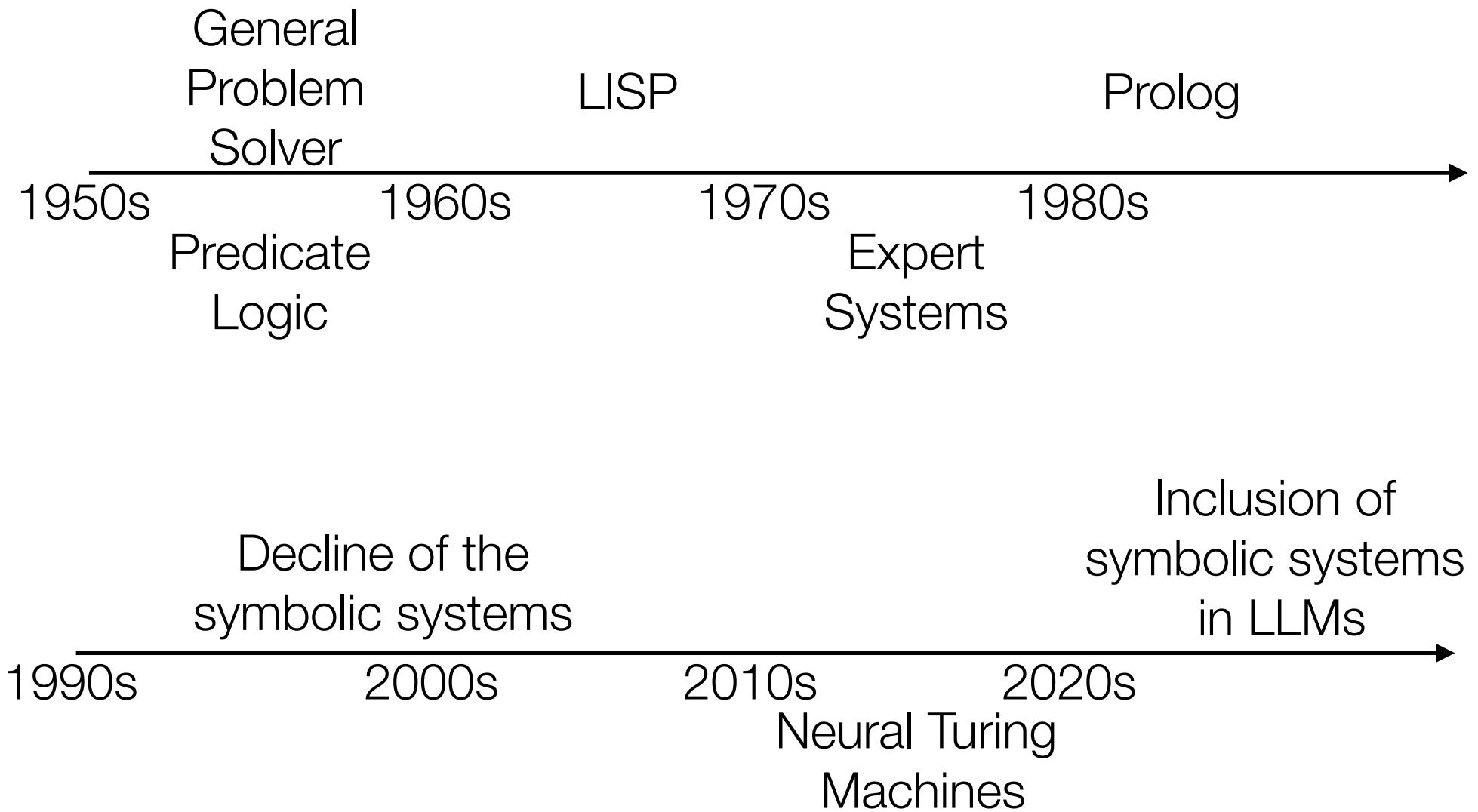
¿Qué pasó entre 1956 y 2018?



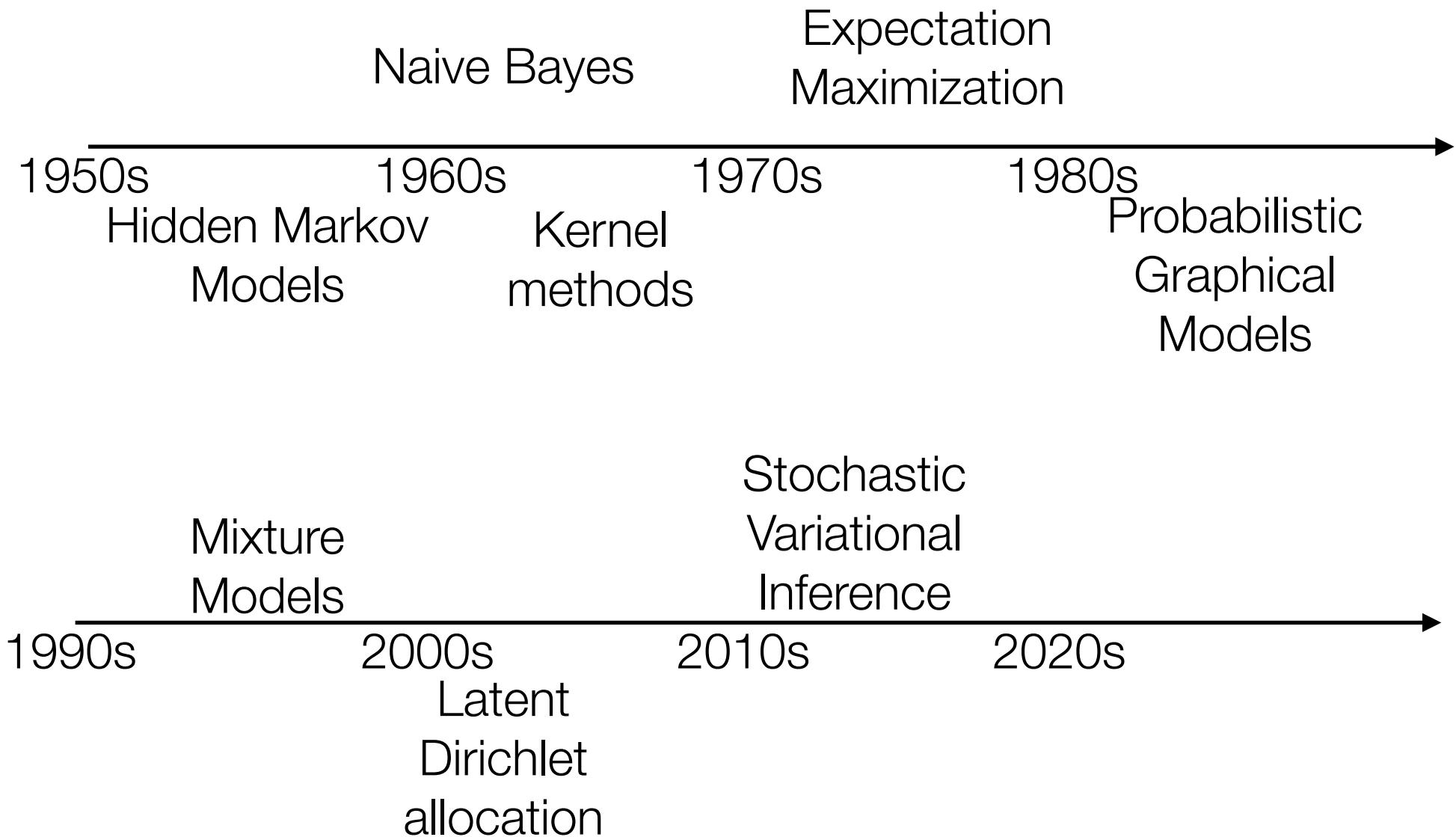
IA, métodos heurísticos



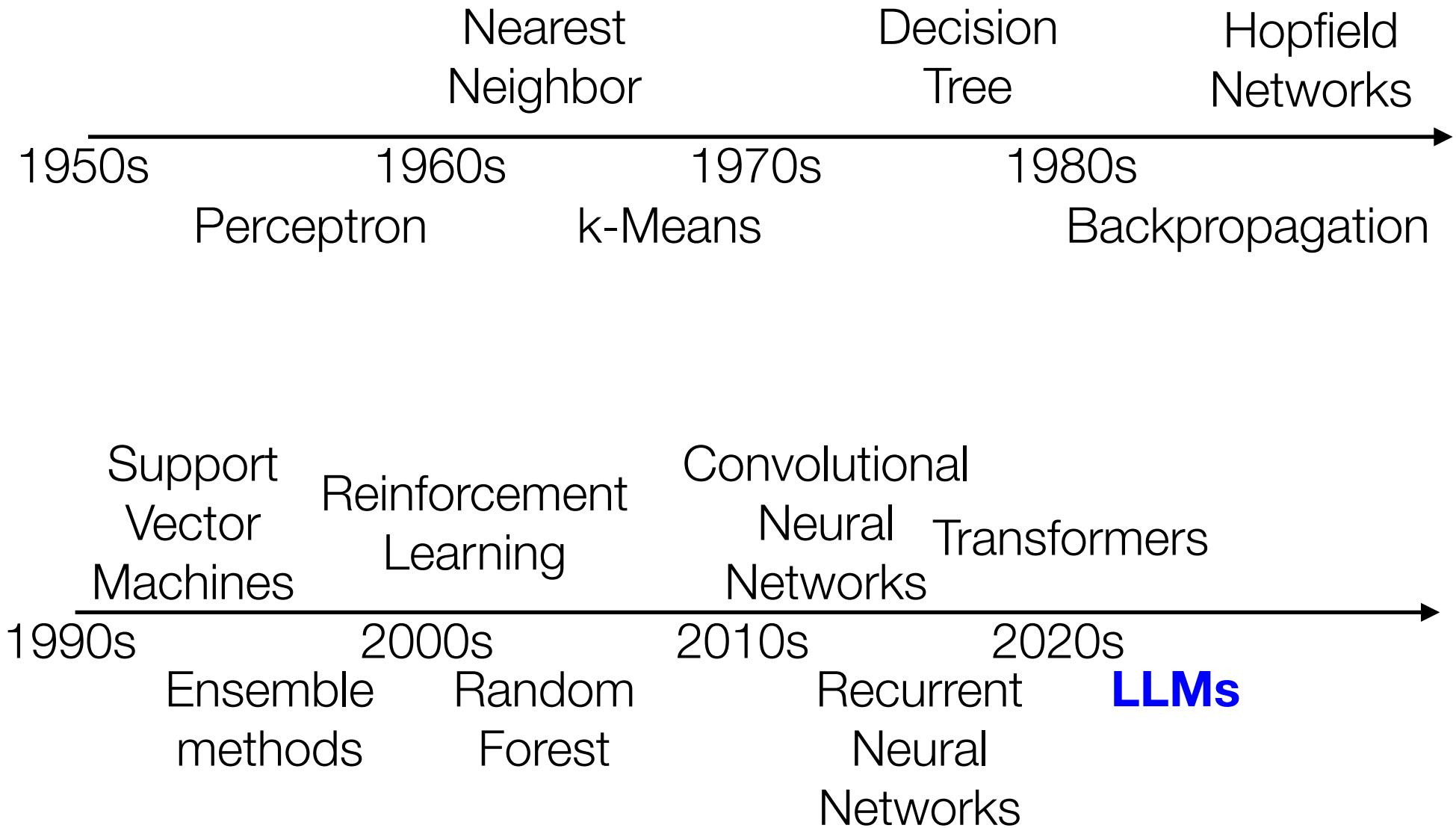
IA, métodos simbólicos (reglas?)



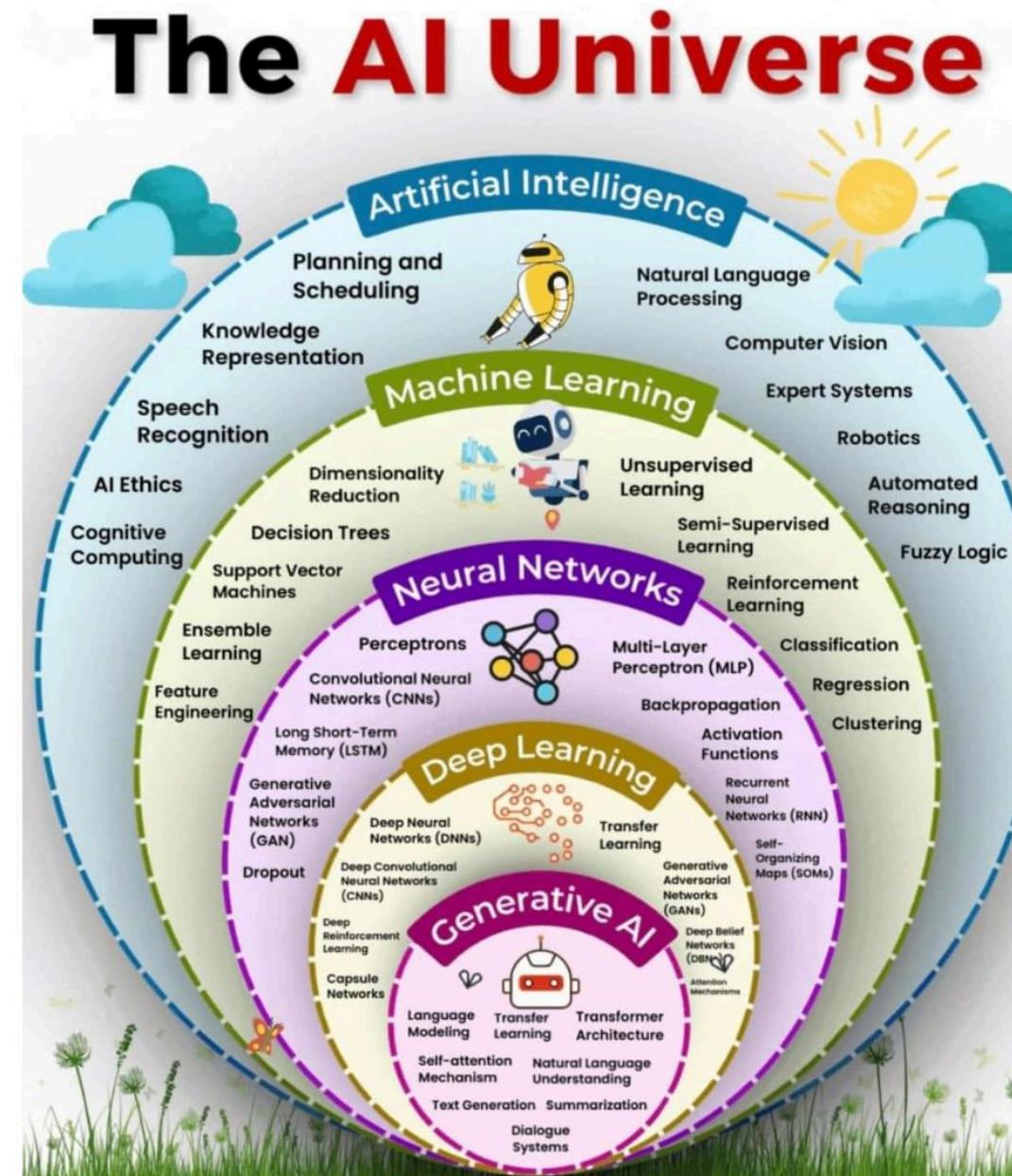
IA, métodos estadísticos



IA, métodos de machine learning



Sin embargo la definición de IA varía mucho



Mito 2:

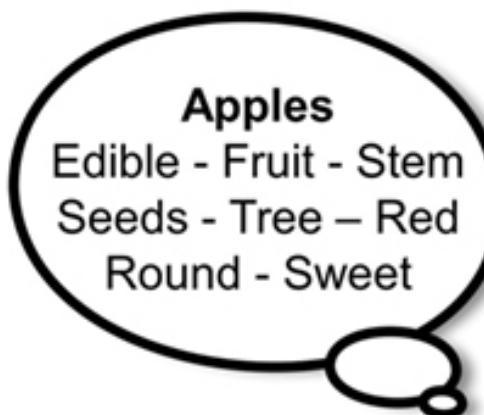
Los LLMs entienden las palabras igual que los humanos

FALSO.

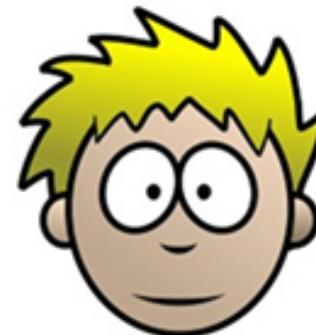
¿Cómo aprendemos las palabras?

- No hay una teoría clara de como aprendemos las palabras, pero el estudio de la memoria semántica reconoce que conceptos similares se agrupan entre si.
- Sin embargo, la activación cerebral de una palabra concreta como por ejemplo “mesa” es muy distinta a la activación de una palabra abstracta como “democracia”.

Semantic Memory



object knowledge learned over many interactions



Episodic Memory



memory for specific events that you have experienced

Los computadores solo trabajan con números

One-hot Encoding

request
feature
how
issue
credit

request →

1	0	0	0	0
---	---	---	---	---

feature →

0	1	0	0	0
---	---	---	---	---

how →

0	0	1	0	0
---	---	---	---	---

issue →

0	0	0	1	0
---	---	---	---	---

credit →

0	0	0	0	1
---	---	---	---	---

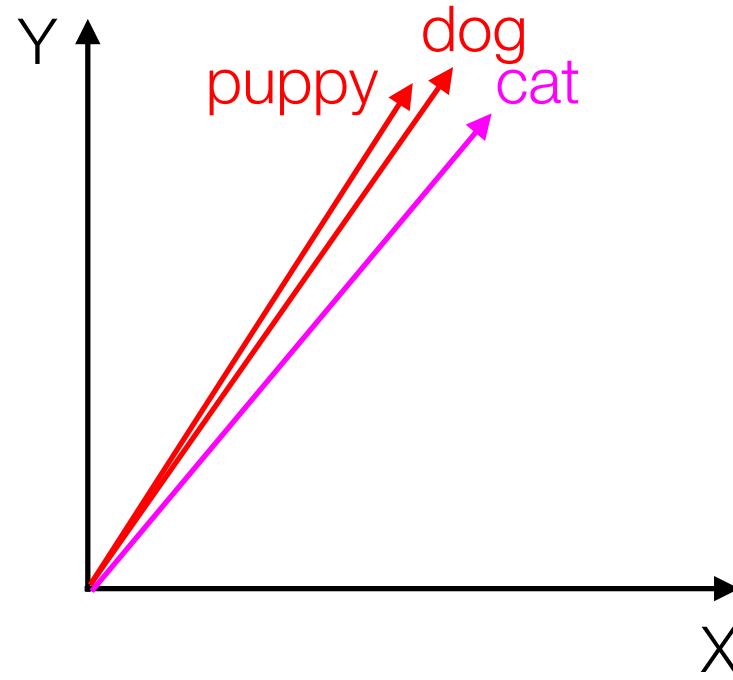
Word Embedding

Cat →	1.2	-0.1	4.3	3.2
Mat →	0.4	2.5	-0.9	0.5
on →	2.1	0.3	0.1	0.4

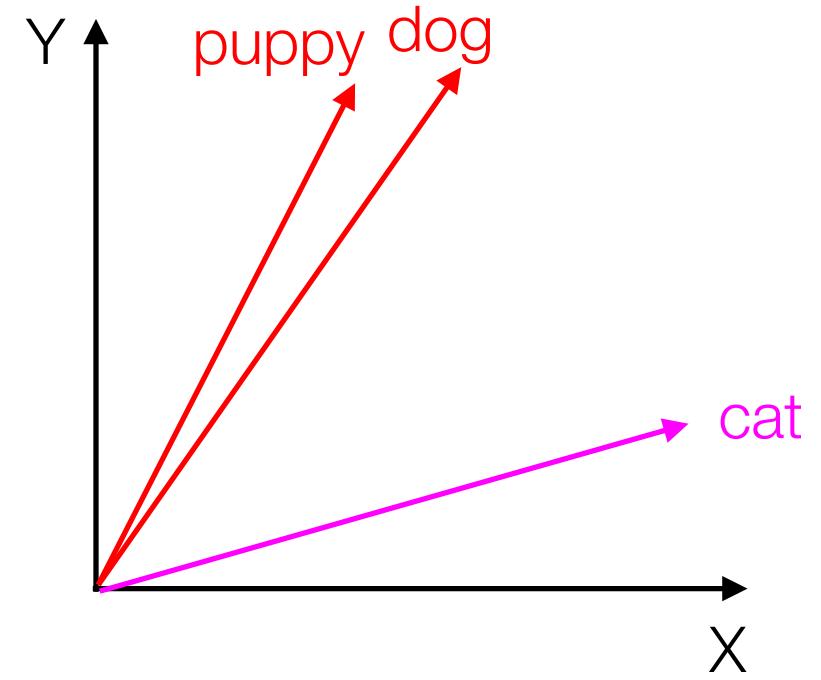
Embedding

- Un embedding es una transformación matemática de datos complejos (palabras, frases, imágenes, etc) a un vector numérico.

$$f(\mathbf{x}) \rightarrow \mathbb{R}^m$$



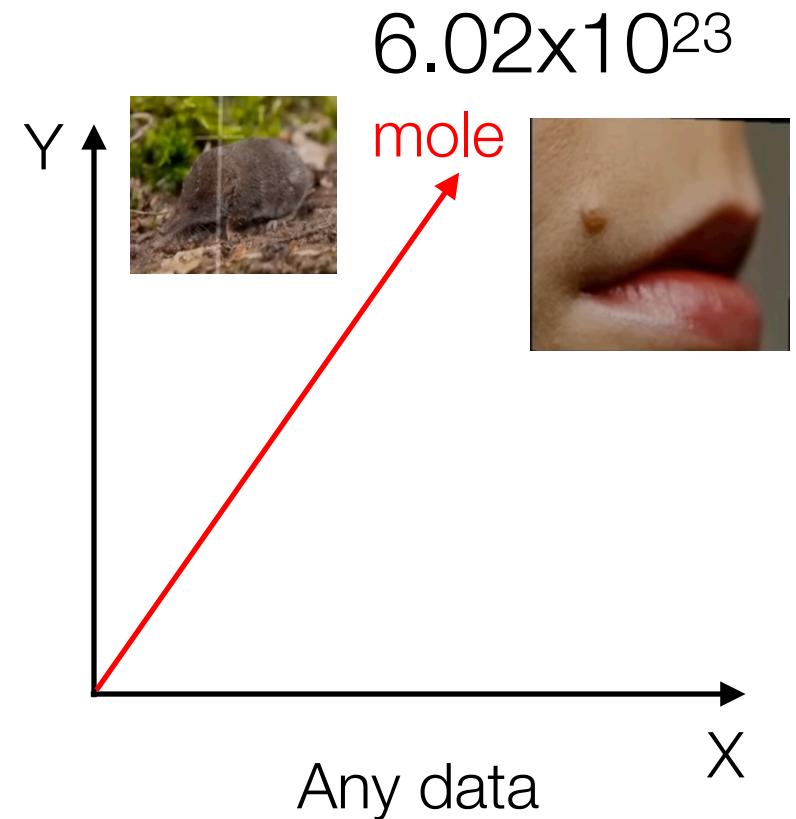
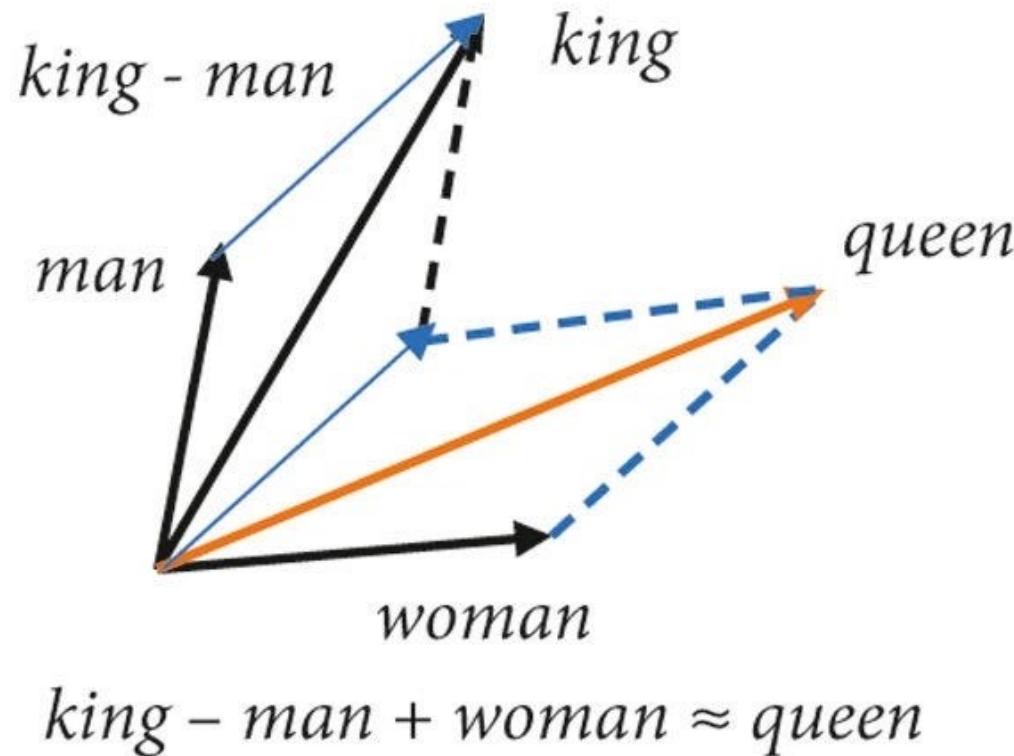
Wikipedia data



Biology data

Embedding, Word2Vec

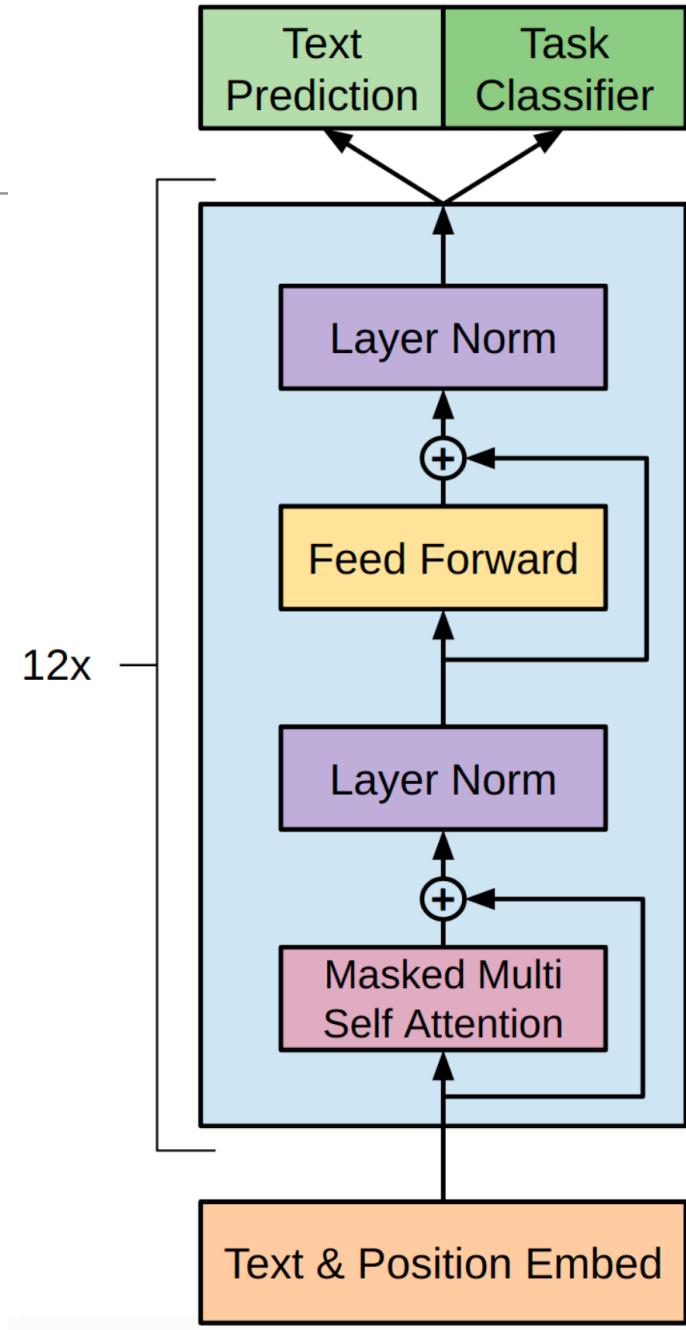
- Word2Vec uno de los más importantes embeddings (previo a los LLMs), aprende relaciones tanto semánticas (agrupa palabras con significado similar) como sintácticas (función gramatical).
- Palabras homónimas tienen el mismo vector.



Arquitectura de los LLMs

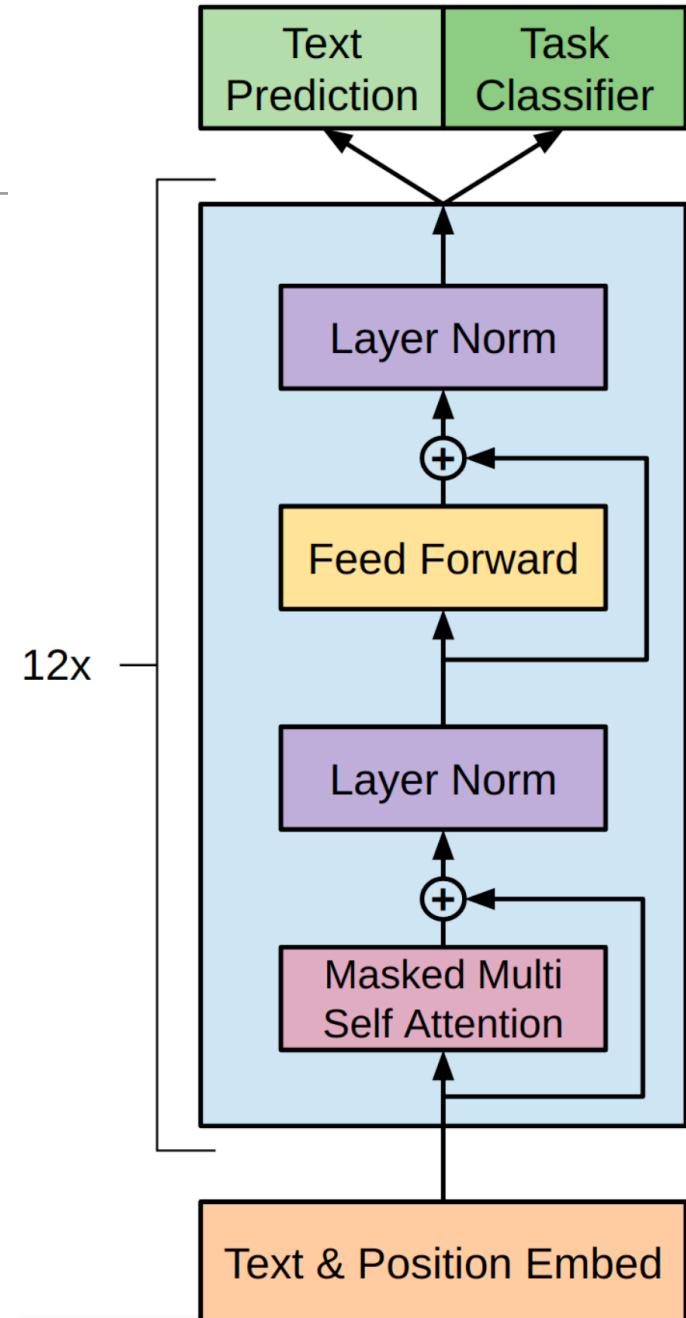
- En machine learning todo se trabaja con números.
- En la arquitectura de un LLM lo primero que se realiza es el embedding de cada una de las palabras en m dimensiones.
- Para gpt 5.0, se estima que $m = 3072$.

Word Embedding				
	1.2	-0.1	4.3	3.2
Cat →	1.2	-0.1	4.3	3.2
Mat →	0.4	2.5	-0.9	0.5
on →	2.1	0.3	0.1	0.4



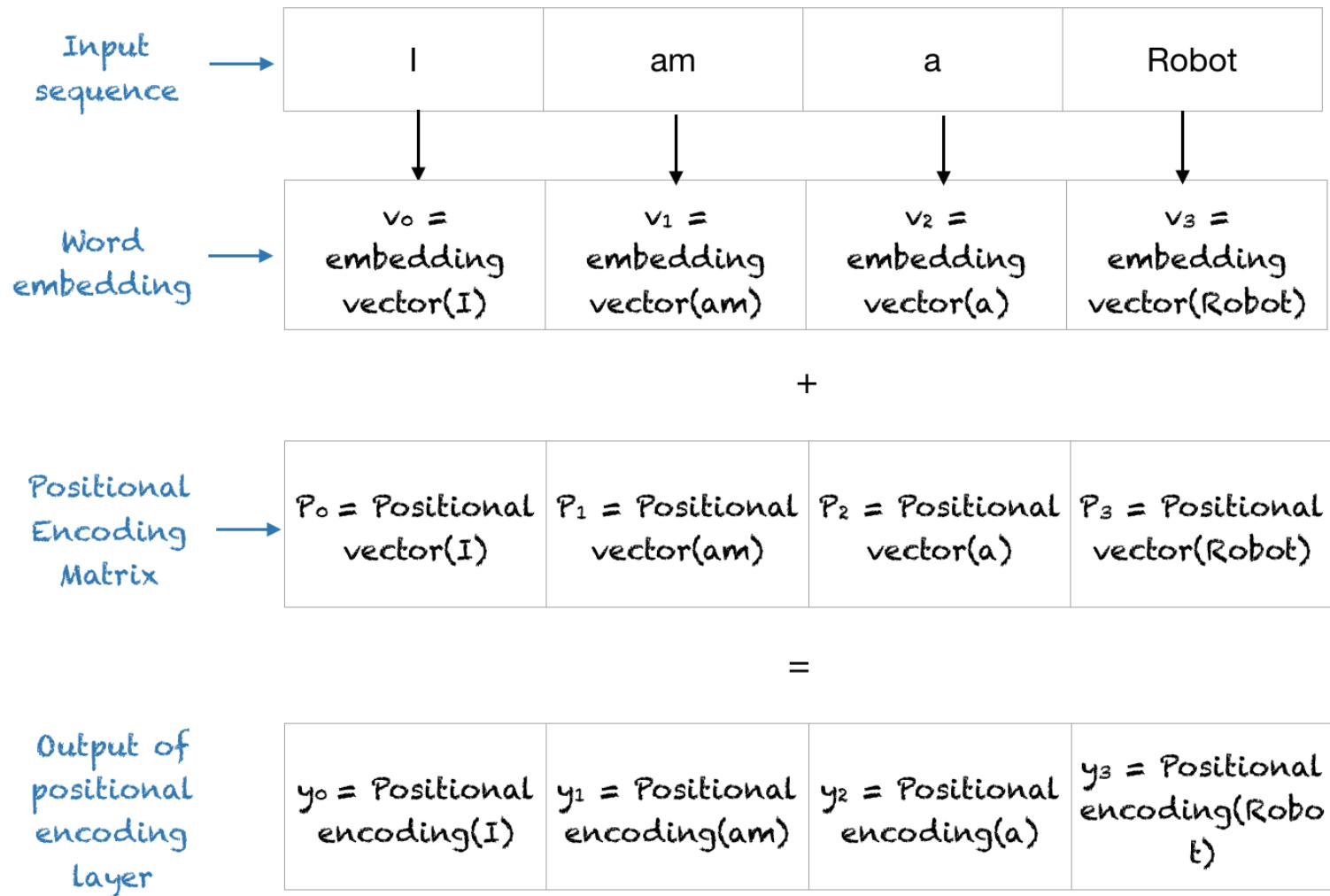
Arquitectura de los LLMs

- En machine learning todo se trabaja con números.
- En la arquitectura de un LLM lo primero que se realiza es el embedding de cada una de las palabras en m dimensiones
- Para gpt 5.0, se estima que $m = 3072$.
- Los LLMs aprenden su propio embedding transformando cada palabra a un vector de dimensión m .
- Por lo cual si escribimos una frase de 10 palabras, por ahora asumamos que se genera un “tensor” de $10 \times m$.



Ejemplo de embedding

- Ejemplo de machinelearningmastery.com, input: “I am a robot”



Mito 3:
Los LLMs es el punto de entrada
para la Inteligencia artificial general

FALSO.

Inteligencia Artificial General (IAG)

- La IAG se refiere a un tipo de IA capaz de comprender, aprender y realizar cualquier tarea intelectual que un humano pueda.
- Actualmente los LLMs no son capaces de generalizar ni siquiera cosas básicas como por ejemplo las sumas o conteo de letras.



Apologies for the repeated errors. Let's recalculate the sum of the frequency correctly:

Sum of Frequency = 125 + 250 + 500 + 750 + 1000 + 1500 + 2000 + 1800 + 1600 + 1400 +
1200 + 1000 + 900 + 800 + 700 + 500 + 400 + 300 + 200 + 150 + 100 + 80 + 60 + 40 + 20
= 17,675

Therefore, the correct sum of the frequency count is 17,675. I apologize for the confusion caused by the earlier incorrect responses.



the answer should be 17375



ChatGPT Error >



How many 'r' in 'strawberry'

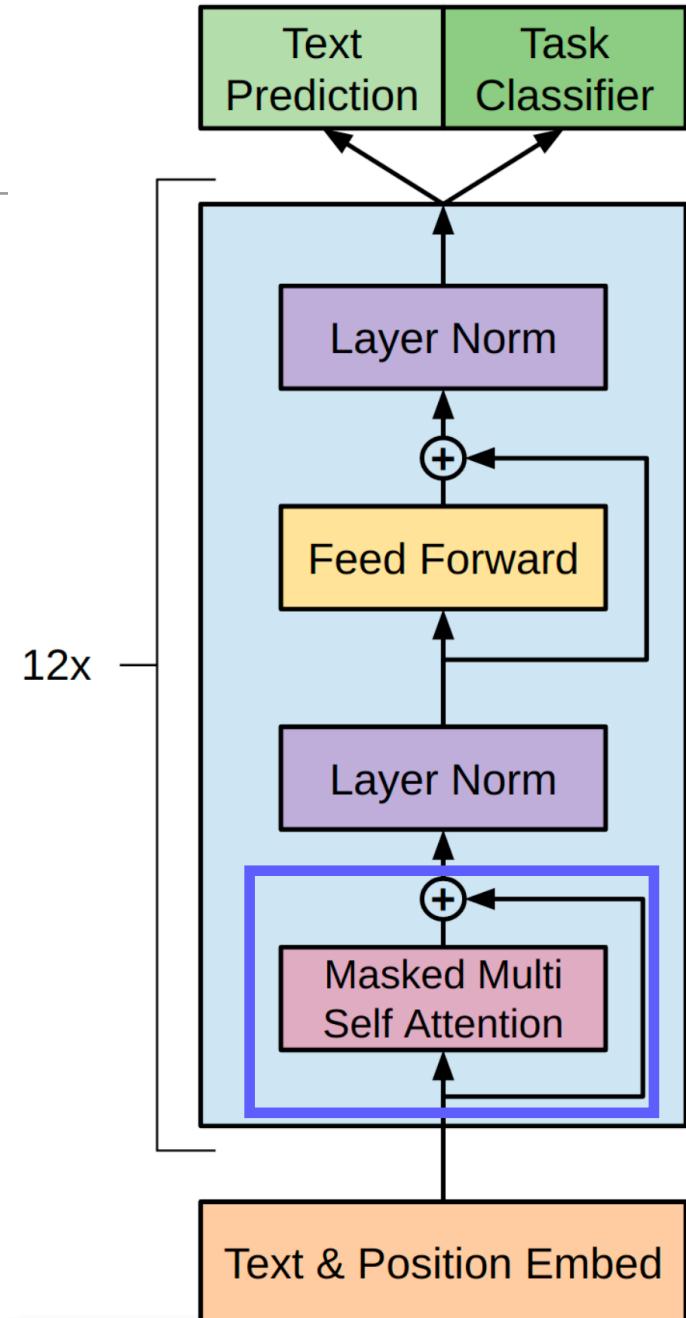


ChatGPT

There are two 'r' characters in the word 'strawberry'.

¿Cómo piensan los LLMs?

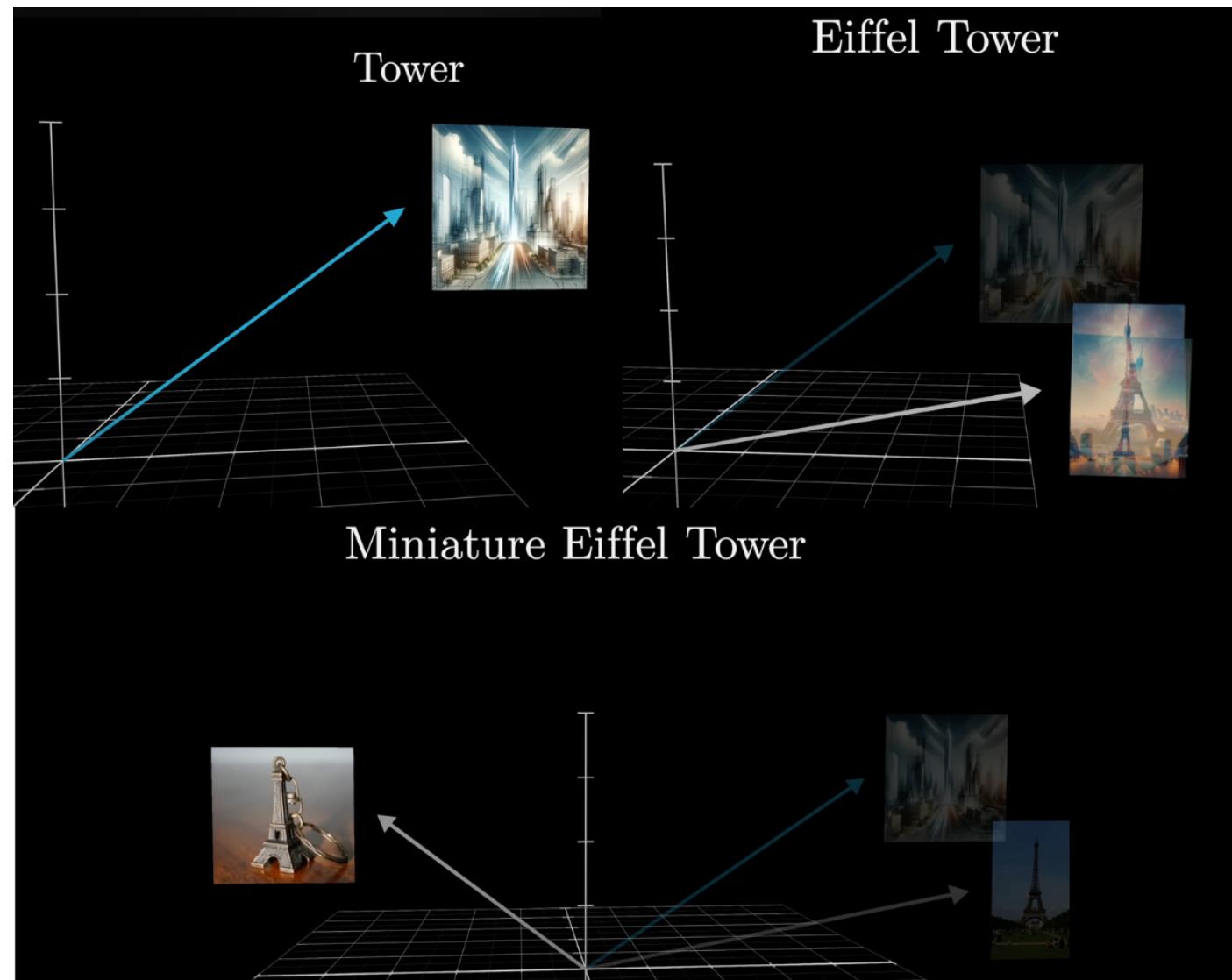
- Recordemos, la frase que nos entregaron se convirtió en un tensor bi-dimensional de $L \times m$.
- Ahora el LLM deberá considerar el contexto de las palabras para modificar estos vectores.
- Esto se realiza en dos pasos, se crea un nuevo tensor bi-dimensional de $L \times m$ y se suma al tensor original. En palabras sencillas, estamos modificando el vector de cada palabra con las siguientes palabras.
- Finalmente, con el vector modificado se predice la siguiente la palabra de respuesta y se vuelve a repetir todo el proceso.



¿Cómo piensan los LLMs?, ejemplo

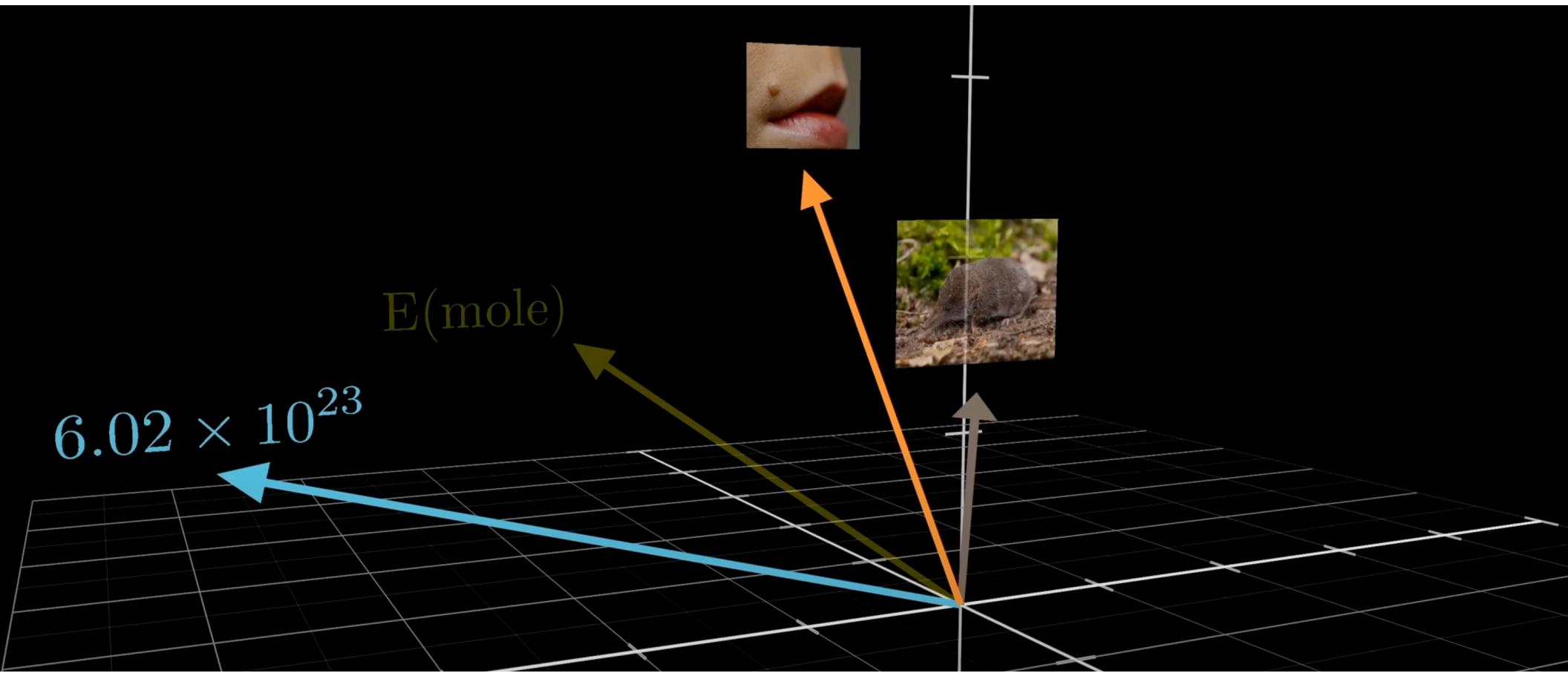
- Ejemplo de 3Blue1Brown.
- El vector tower es modificado por Eiffel y posteriormente modificado de nuevo por Miniature.
- Ojo, el vector de cada palabra es modificado.

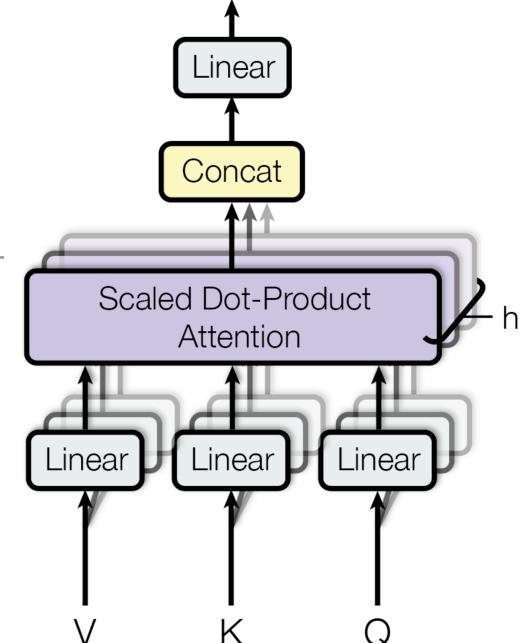
Cambios del vector Tower



¿Cómo piensan los LLMs?, ejemplo

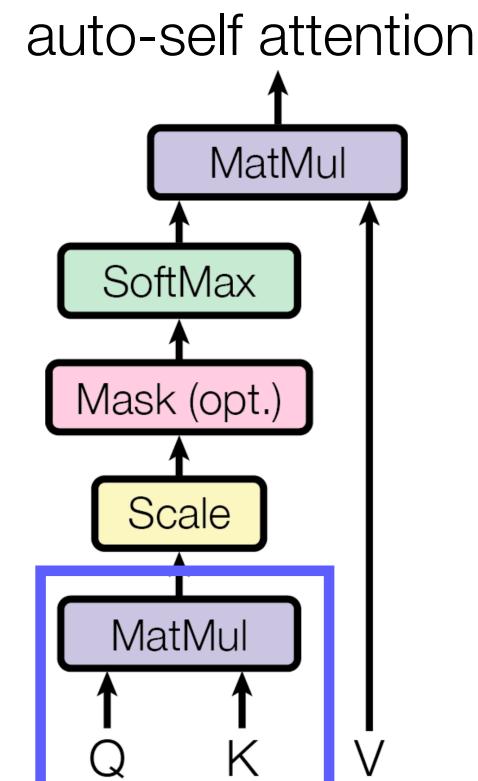
- Ejemplo de 3Blue1Brown.
- Este proceso de atención nos permitirá separar palabras con el mismo significado según su contexto.





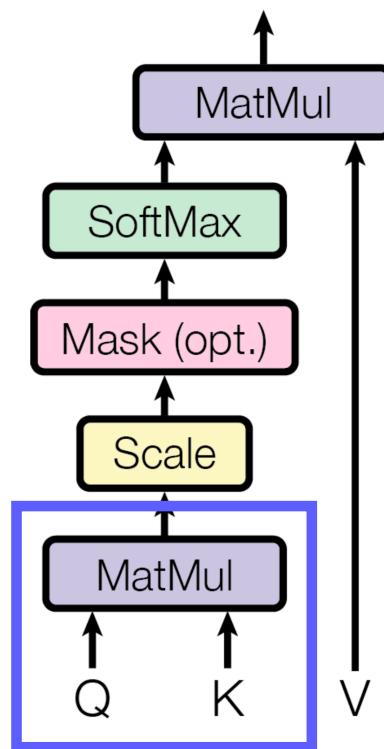
Multi-Head attention

- En el multi-head attention, tengo h procesos en paralelo cada uno modificará en algo cada vector codificado.
- Cada proceso se llama auto-self attention, y las entradas son Query (**Q**), Key (**K**), y Value (**V**):
 $\mathbf{Q} = \mathbf{E} \times \mathbf{W}_Q$ (weights of size $d \times d_k$).
 $\mathbf{K} = \mathbf{E} \times \mathbf{W}_K$ (weights of size $d \times d_k$).
 $\mathbf{V} = \mathbf{E} \times \mathbf{W}_V$ (weights of size $d \times d_v$).
 Donde las matrices **W** son pesos a aprender.
- La idea principal es que el modelo aprenda que palabra puede afectar, permitiendo modificar el vector original.

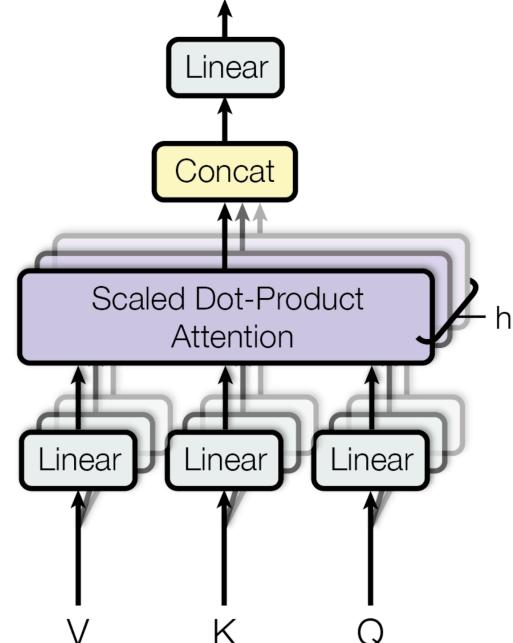


Auto-self attention, ejemplo

- Ejemplo de 3Blue1Brown, el sustantivo “creature” se ve afectado por los adjetivos “fluffy” y “blue”.

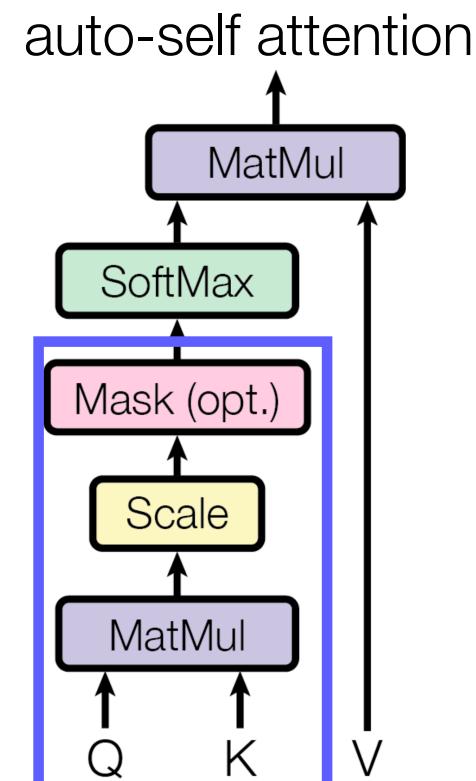


	a	fluffy	blue	creature	roamed	the	verdant
	\vec{E}_1	\vec{E}_2	\vec{E}_3	\vec{E}_4	\vec{E}_5	\vec{E}_6	\vec{E}_7
	$\downarrow W_Q$						
$a \rightarrow \vec{E}_1 \xrightarrow{W_k} \vec{K}_1$	+0.7	-83.7	-24.7	-27.8	-5.2	-89.3	-45.2
$\text{fluffy} \rightarrow \vec{E}_2 \xrightarrow{W_k} \vec{K}_2$	-73.4	+2.9	-5.4	+93.0	-48.2	-87.3	-49.7
$\text{blue} \rightarrow \vec{E}_3 \xrightarrow{W_k} \vec{K}_3$	-53.4	-5.7	+1.8	+93.4	-55.6	-56.0	-26.1
$\text{creature} \rightarrow \vec{E}_4 \xrightarrow{W_k} \vec{K}_4$	-21.5	-29.7	-56.1	+4.9	-32.4	-92.3	-9.5
$\text{roamed} \rightarrow \vec{E}_5 \xrightarrow{W_k} \vec{K}_5$	-20.1	-40.9	-87.8	-55.4	+0.6	-64.7	-96.7
$\text{the} \rightarrow \vec{E}_6 \xrightarrow{W_k} \vec{K}_6$	-87.9	-33.3	-22.6	-31.4	+5.5	+0.6	-4.6
$\text{verdant} \rightarrow \vec{E}_7 \xrightarrow{W_k} \vec{K}_7$	-41.2	-55.5	-42.3	-59.8	-79.0	-97.9	+3.7



Auto-self attention, continuación

- En el multi-head attention, tengo h procesos en paralelo cada uno modificará en algo cada vector codificado.
- Cada proceso se llama auto-self attention, y las entradas son Query (**Q**), Key (**K**), y Value (**V**).
- La idea principal es que el modelo aprenda que palabra puede afectar, permitiendo modificar el vector original.
- La matriz resultante de **Q** y **K** se escala (estabilidad numérica y evitar highly concentrated distribution en el módulo softmax) y se aplica una máscara (los LLMs producen palabra por palabra, por lo cual no deben ver a futuro la generación de palabras).



Predicción de palabra por palabra

Binge ... on | - | and | of | is

Binge **drinking** ... is | and | had | in | was

Binge drinking **may** ... be | also | have | not | increase

Binge drinking may **not** ... be | have | cause | always | help

Binge drinking may not **necessarily** ... be | lead | cause | results | have

Binge drinking may not necessarily **kill** ... you | the | a | people | your

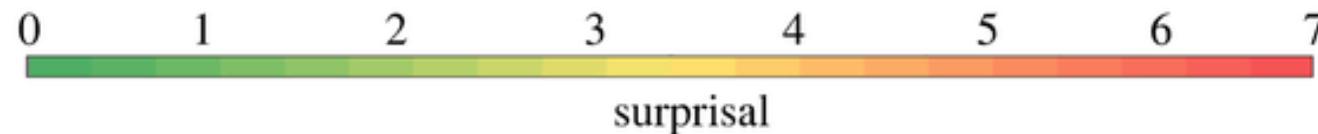
Binge drinking may not necessarily kill **or** ... even | injure | kill | cause | prevent

Binge drinking may not necessarily kill or **even** ... kill | prevent | cause | reduce | injure

Binge drinking may not necessarily kill or even **damage** ... your | the | a | you | someone

Binge drinking may not necessarily kill or even damage **brain** ... cells | functions | tissue | neurons

Binge drinking may not necessarily kill or even damage brain **cells**, ... some | it | the | is | long

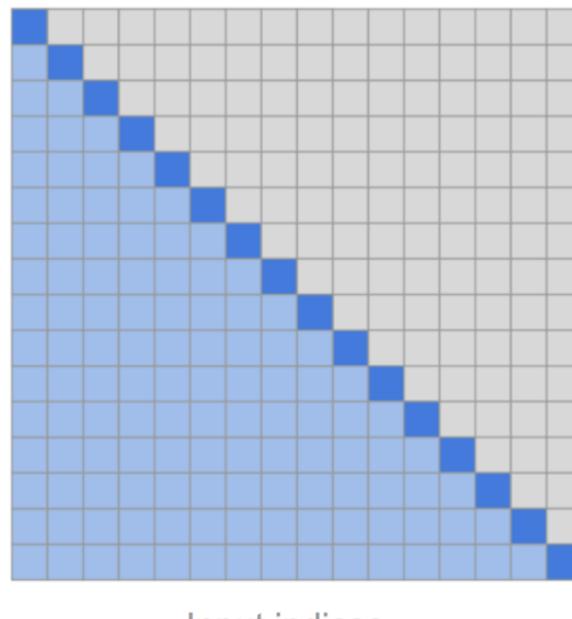


Prediction as a basis for skilled reading: Insights from modern language models, <https://doi.org/10.1098/rsos.211837>.

Auto-self attention, ejemplo de máscara

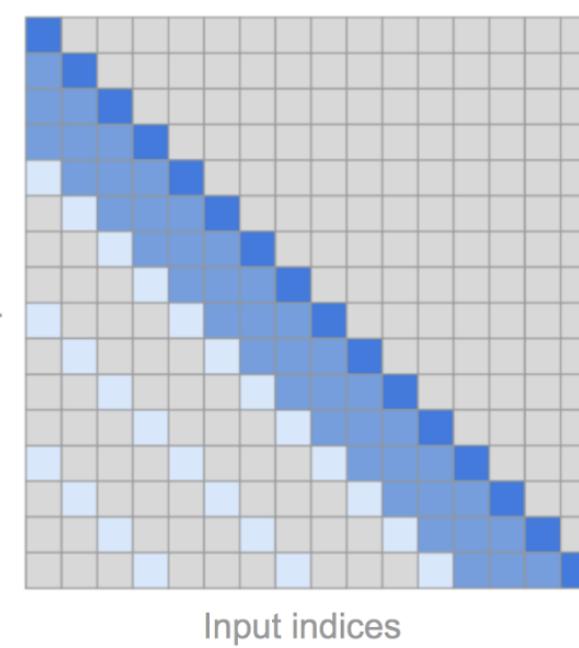
- Ejemplo de Lil'Log, distintos tipos de atención para evitar que palabras futuras afectan la predicción de la siguiente palabra.

Self-attention connectivity matrix



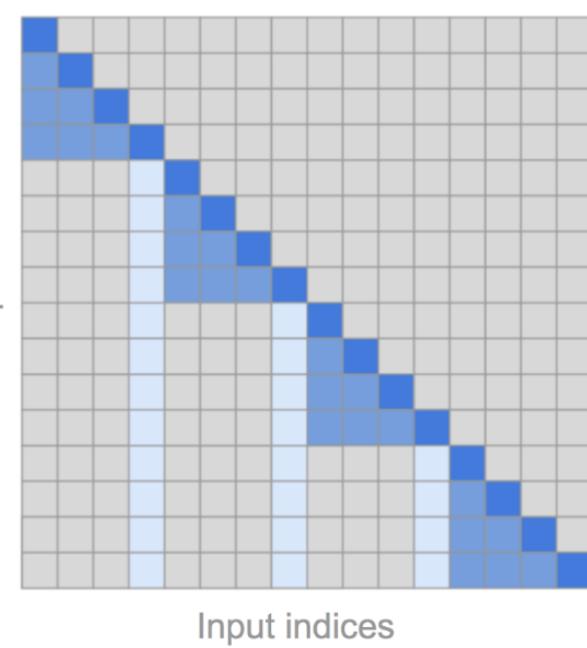
(a) Transformer

Output indices



(b) Sparse Transformer with
strided attention.

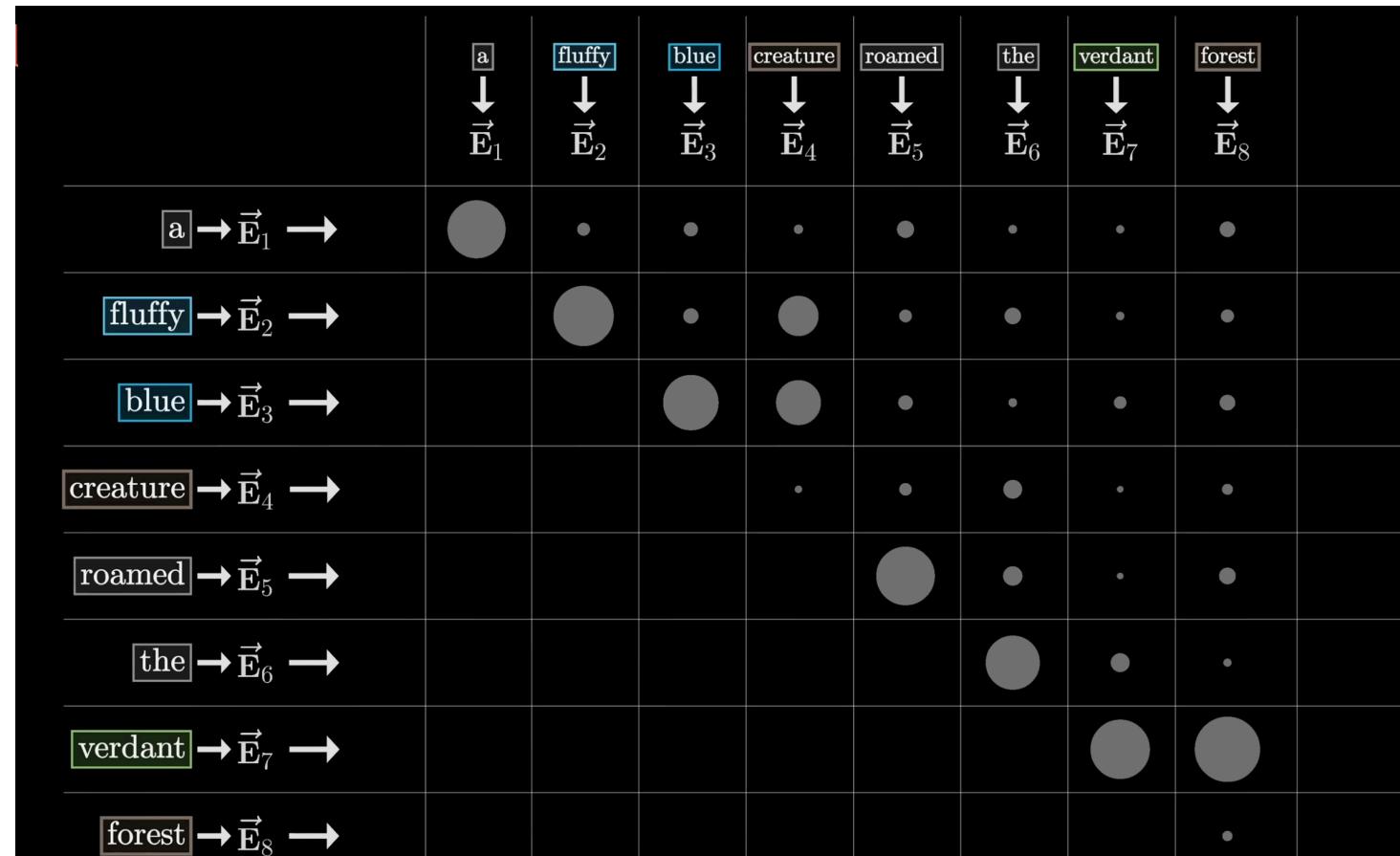
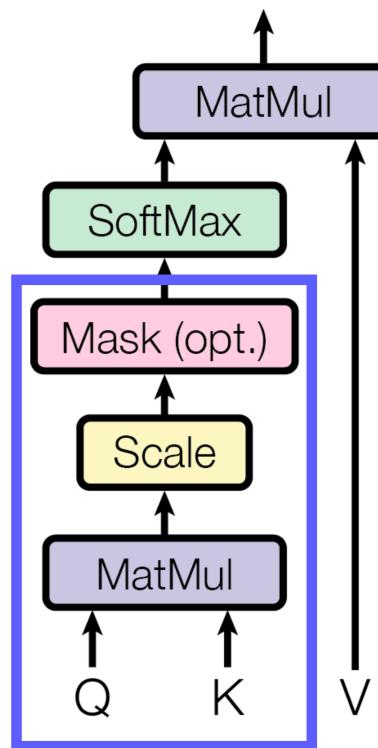
Output indices



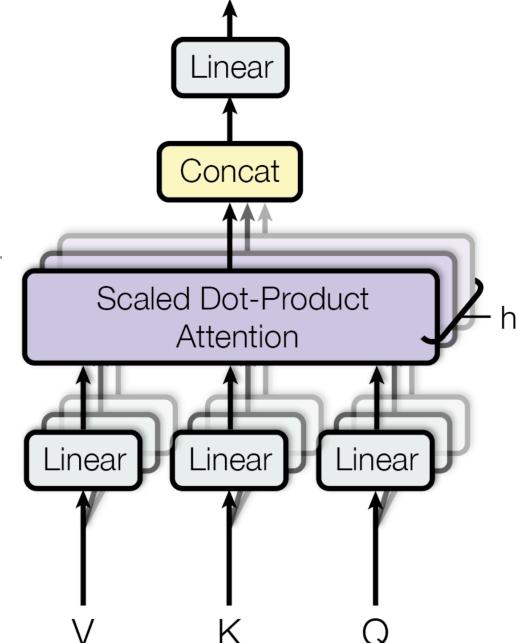
(c) Sparse Transformer with
fixed attention.

Auto-self attention, ejemplo

- Ejemplo de 3Blue1Brown, el sustantivo “creature” se ve afectado por los adjetivos “fluffy” y “blue”.

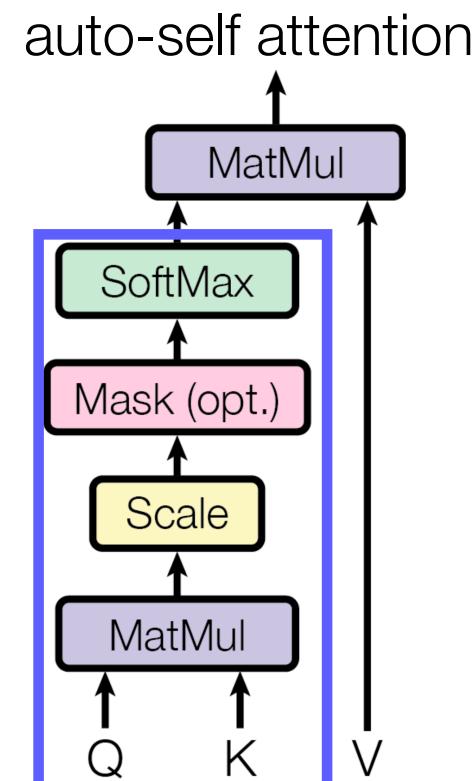


- Atención, por alguna razón 3Blue1Brown lo explica como K^*Q , en evz de Q^*K , por eso el orden de la matriz.



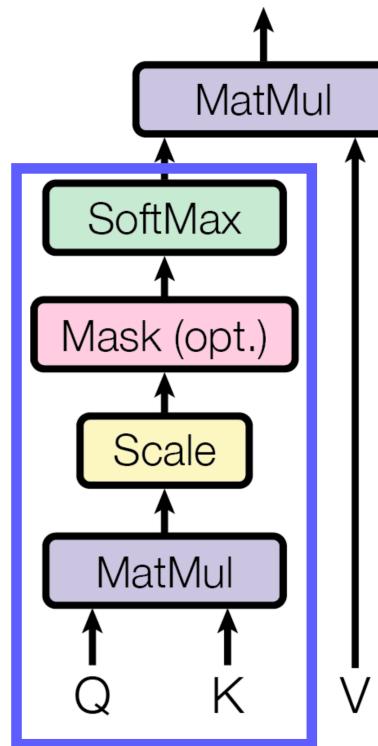
Auto-self attention, continuación

- En el multi-head attention, tengo h procesos en paralelo cada uno modificará en algo cada vector codificado.
- Cada proceso se llama auto-self attention, y las entradas son Query (**Q**), Key (**K**), y Value (**V**).
- La idea principal es que el modelo aprenda que palabra puede afectar, permitiendo modificar el vector original.
- La matriz resultante de **Q** y **K** se escala y se aplica una máscara (los LLMs producen palabra por palabra, por lo cual no deben ver a futuro la generación de palabras).
- Luego se calcula la “influencia” de cada palabra aplicando la función Softmax.



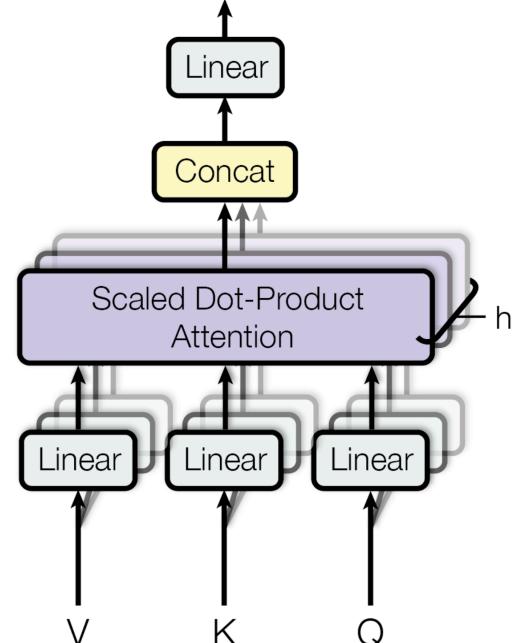
Auto-self attention, ejemplo

- Ejemplo de 3Blue1Brown, “creature” se ve afectado solo por palabras anteriores.



	\vec{E}_1	\vec{E}_2	\vec{E}_3	\vec{E}_4	\vec{E}_5	\vec{E}_6	\vec{E}_7	\vec{E}_8
a	0.00							
fluffy		0.42						
blue			0.58					
creature				0.00				
roamed					0.00			
the						0.00		
verdant							0.00	
forest								0.00

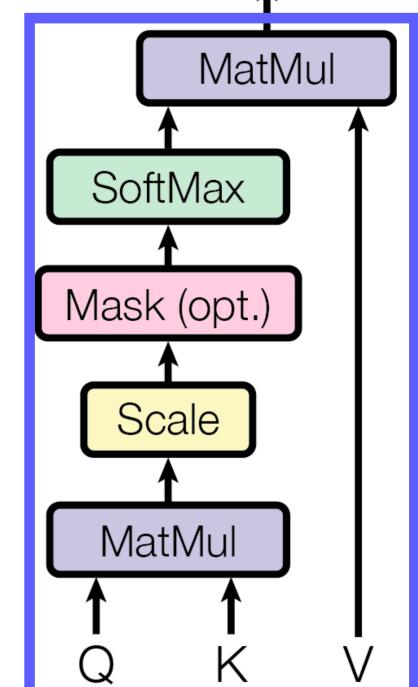
- Atención, por alguna razón 3Blue1Brown lo explica como K^*Q , en vez de Q^*K , por eso el orden de la matriz.



Auto-self attention, continuación

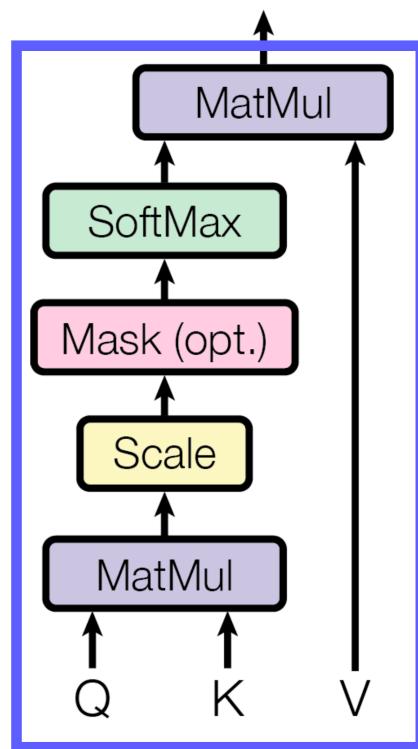
- En el multi-head attention, tengo h procesos en paralelo cada uno modificará en algo cada vector codificado.
- Cada proceso se llama auto-self attention, y las entradas son Query (**Q**), Key (**K**), y Value (**V**).
- La idea principal es que el modelo aprenda que palabra puede afectar, permitiendo modificar el vector original.
- La matriz resultante de **Q** y **K** se escala y se aplica una máscara (los LLMs producen palabra por palabra, por lo cual no deben ver a futuro la generación de palabras).
- Luego se calcula la “influencia” de cada palabra aplicando la función Softmax, y el tensor resultante se multiplica por **V**, dando la importancia de esta atención en los vectores de las palabras.

auto-self attention



Auto-self attention, ejemplo

- Ejemplo de 3Blue1Brown, el sustantivo “creature” se ve afectado por los adjetivos “fluffy” y “blue”.

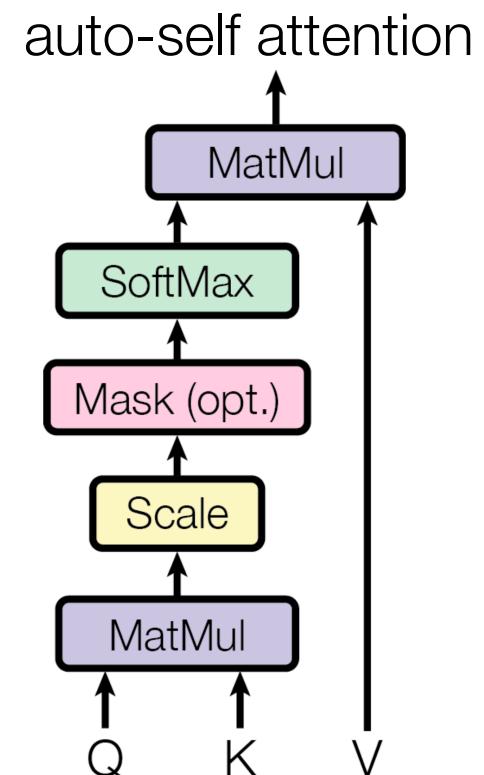
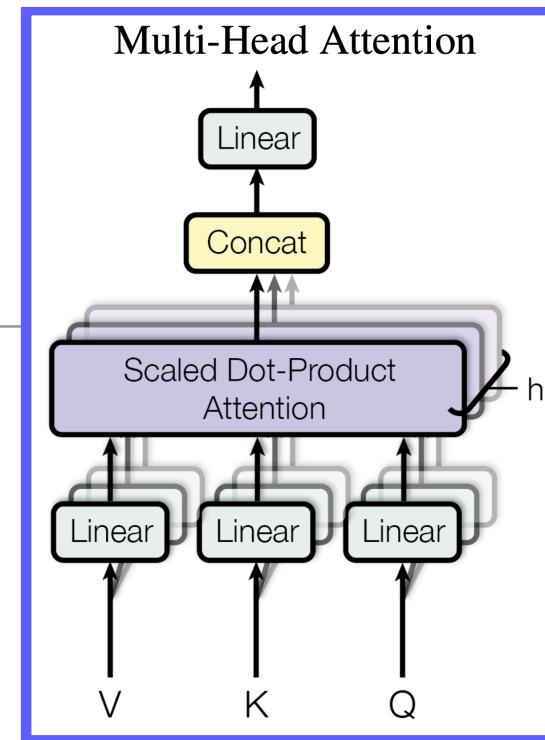


	a	fluffy	blue	creature	roamed	the	verdant	forest
\vec{E}_1	\vec{v}_1							
\vec{E}_2		\vec{v}_2						
\vec{E}_3			\vec{v}_3					
\vec{E}_4				\vec{v}_4				
\vec{E}_5					\vec{v}_5			
\vec{E}_6						\vec{v}_6		
\vec{E}_7							\vec{v}_7	
\vec{E}_8								\vec{v}_8

- Atención, por alguna razón 3Blue1Brown lo explica como K^*Q , en evz de Q^*K , por eso el orden de la matriz.

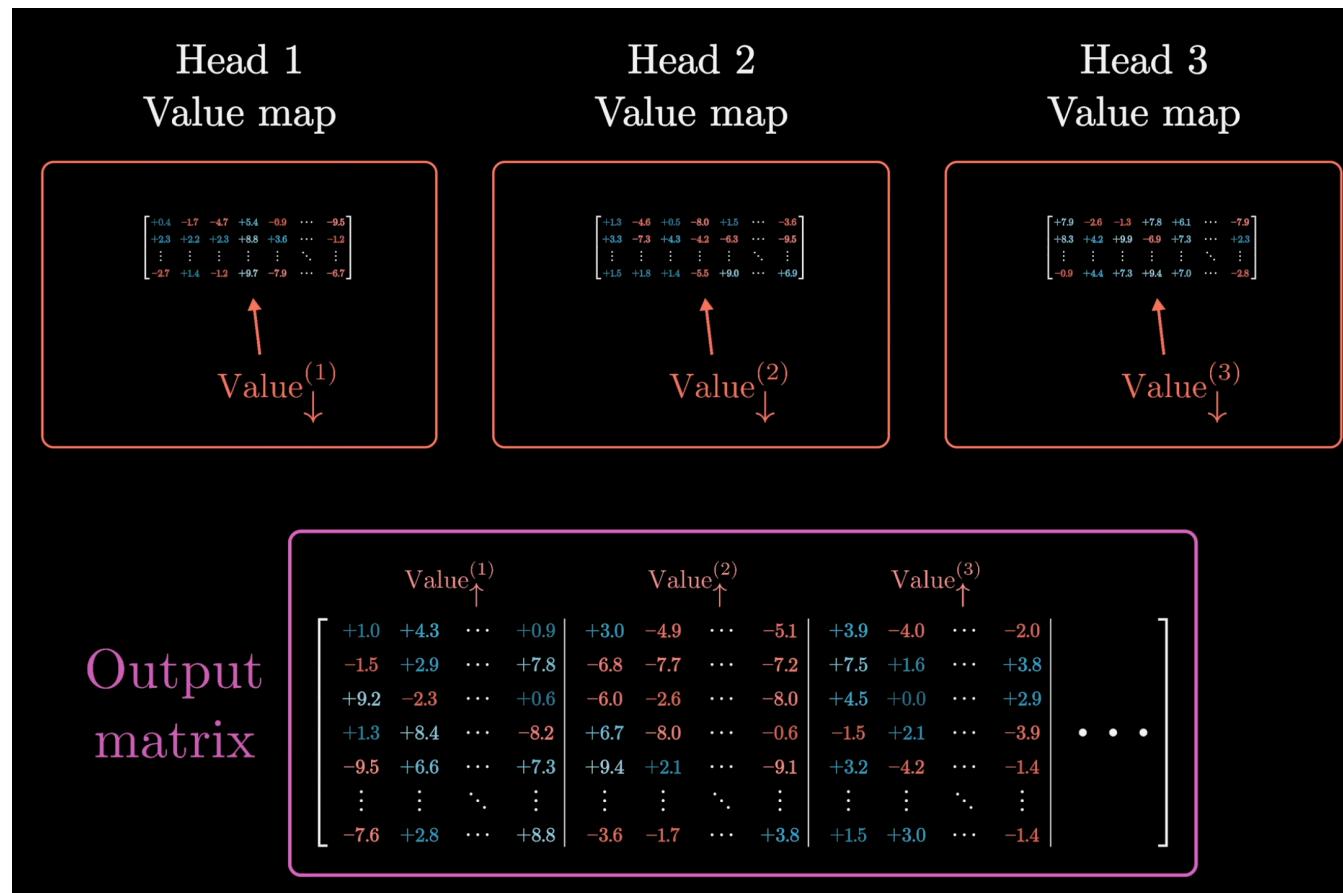
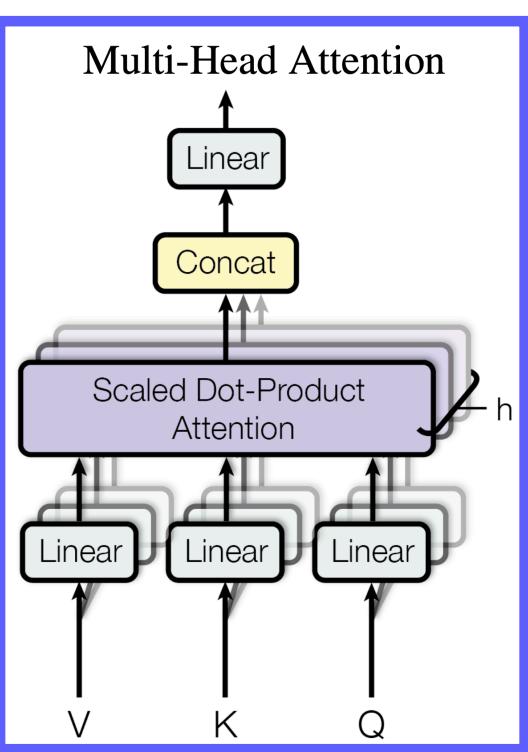
Auto-self attention, continuación

- En el multi-head attention, tengo h procesos en paralelo cada uno modificará en algo cada vector codificado.
- Cada proceso se llama auto-self attention, y las entradas son Query (**Q**), Key (**K**), y Value (**V**).
- La matriz resultante de **Q** y **K** se escala y se aplica una máscara (los LLMs producen palabra por palabra, por lo cual no deben ver a futuro la generación de palabras).
- Luego se calcula la “influencia” de cada palabra aplicando la función Softmax, y el tensor resultante se multiplica por V, dando la importancia de esta atención en los vectores de las palabras.
- Finalmente, se unen las influencias de los h procesos, se combinan a través de una nueva matriz **W** y se agregan al tensor original de las palabras.



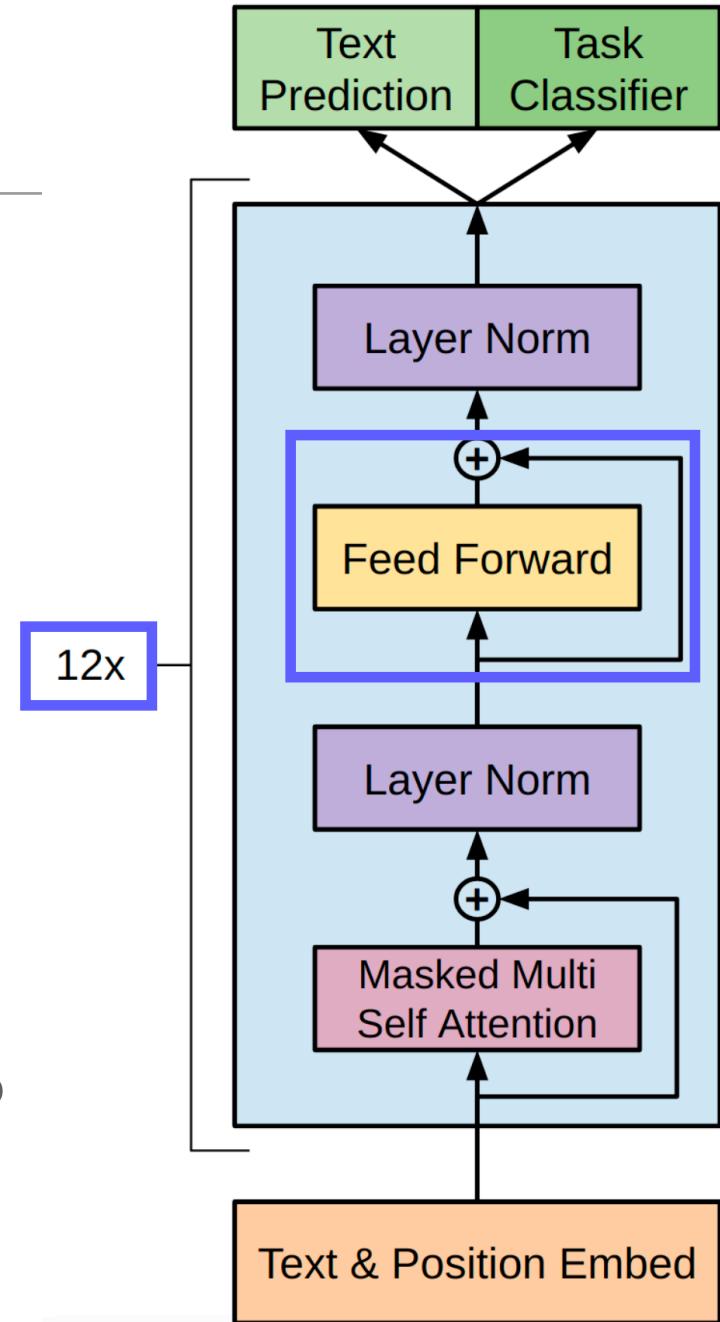
Auto-self attention, ejemplo

- Ejemplo de 3Blue1Brown.



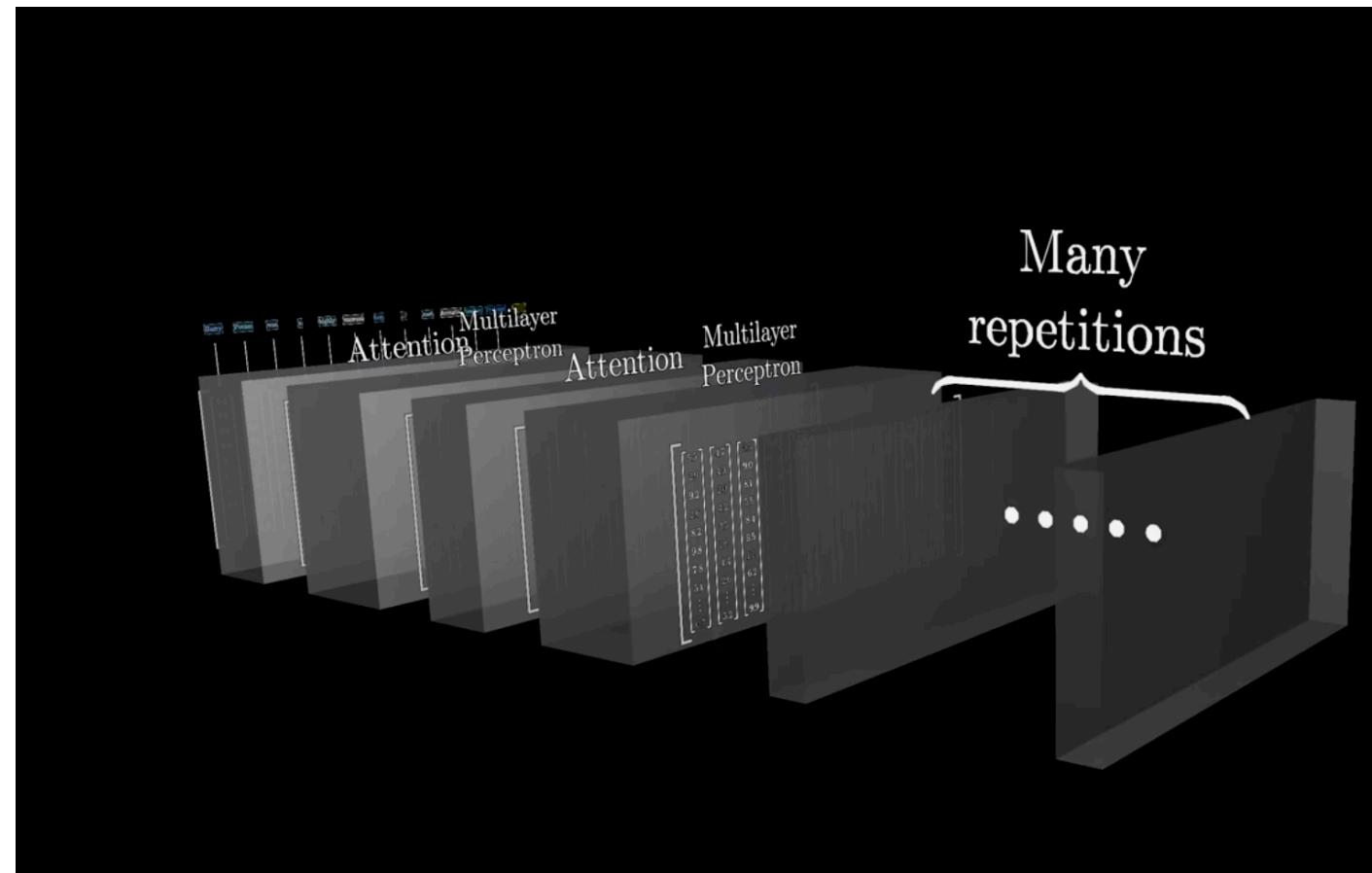
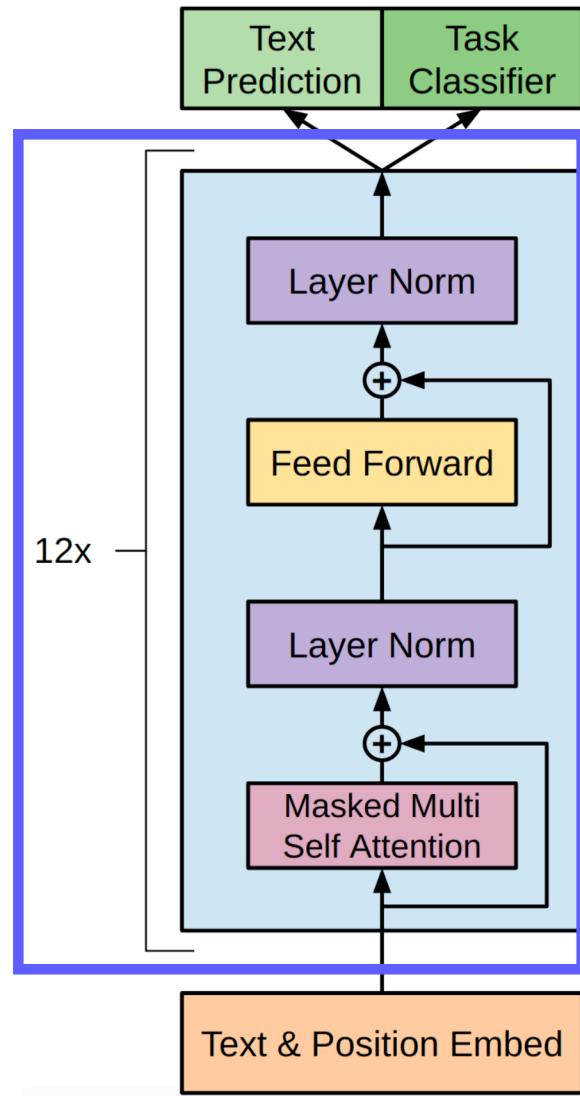
¿Cómo piensan los LLMs?

- Recordemos, la frase que nos entregaron se convirtió en un tensor bi-dimensional de $L \times m$.
- Ahora el LLM deberá considerar el contexto de las palabras para modificar estos vectores (Masked Multi Self Attention).
- Se hace una normalización de los vectores (cada fila suma 1.0) y posteriormente se utiliza una red feed forward para modificar los pesos.
- La “idea” de la red es incrementar la dimensionalidad de los datos (usualmente $4 \cdot m$) para aprender representaciones complejas y luego volver a la dimensión original (m).
- En el caso de GPT1.0, todo este proceso se repite 12 veces para finalmente usar una nueva red feedforward para predecir la siguiente palabra.



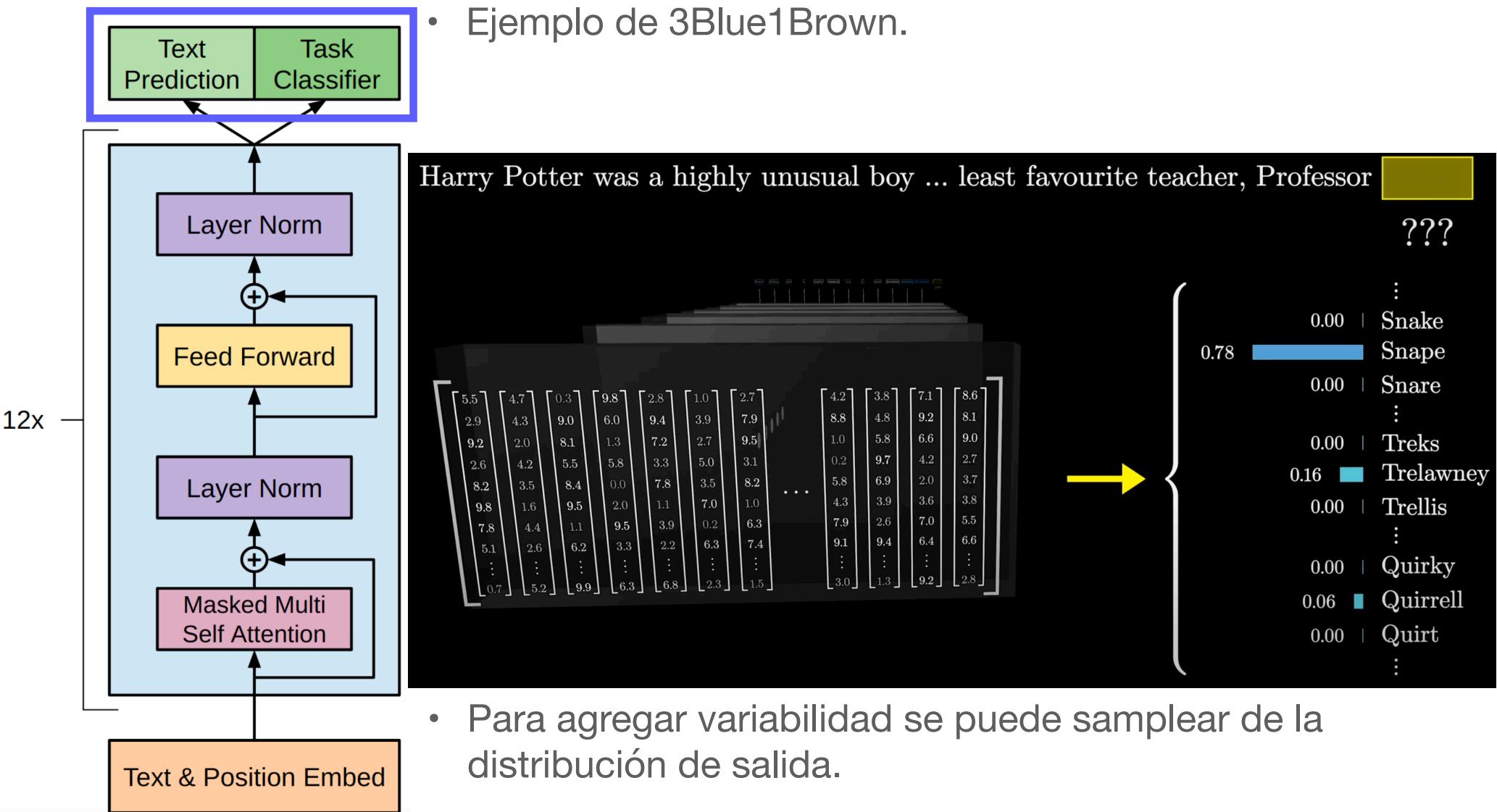
Transformer, ejemplo

- Ejemplo de 3Blue1Brown.



Transformer prediction, ejemplo

- Ejemplo de 3Blue1Brown.



¿Cómo se entrena los LLMs?

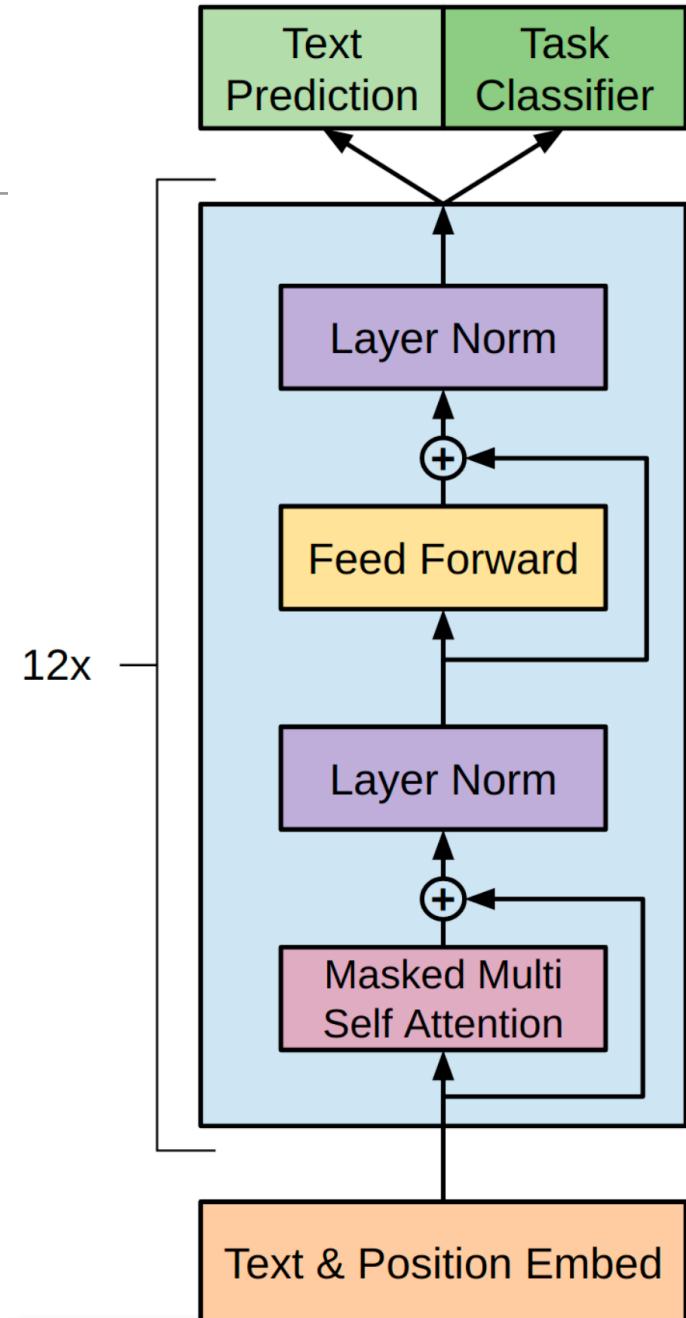
- En total, el modelo GPT 1.0 tiene aproximadamente ~117M parameters.
- The entrenamiento está basado en la predicción de la próxima palabra y se uso BooksCorpus (7,000 libros, ~800 millones de palabras).
- Una simple frase puede utilizar muchas veces en el proceso de entrenamiento. Por ejemplo:

Input: ¿, y= Hola

Input: ¿Hola, y= cómo

Input: ¿Hola, cómo, y= están

Input: ¿Hola, cómo están, y= ?

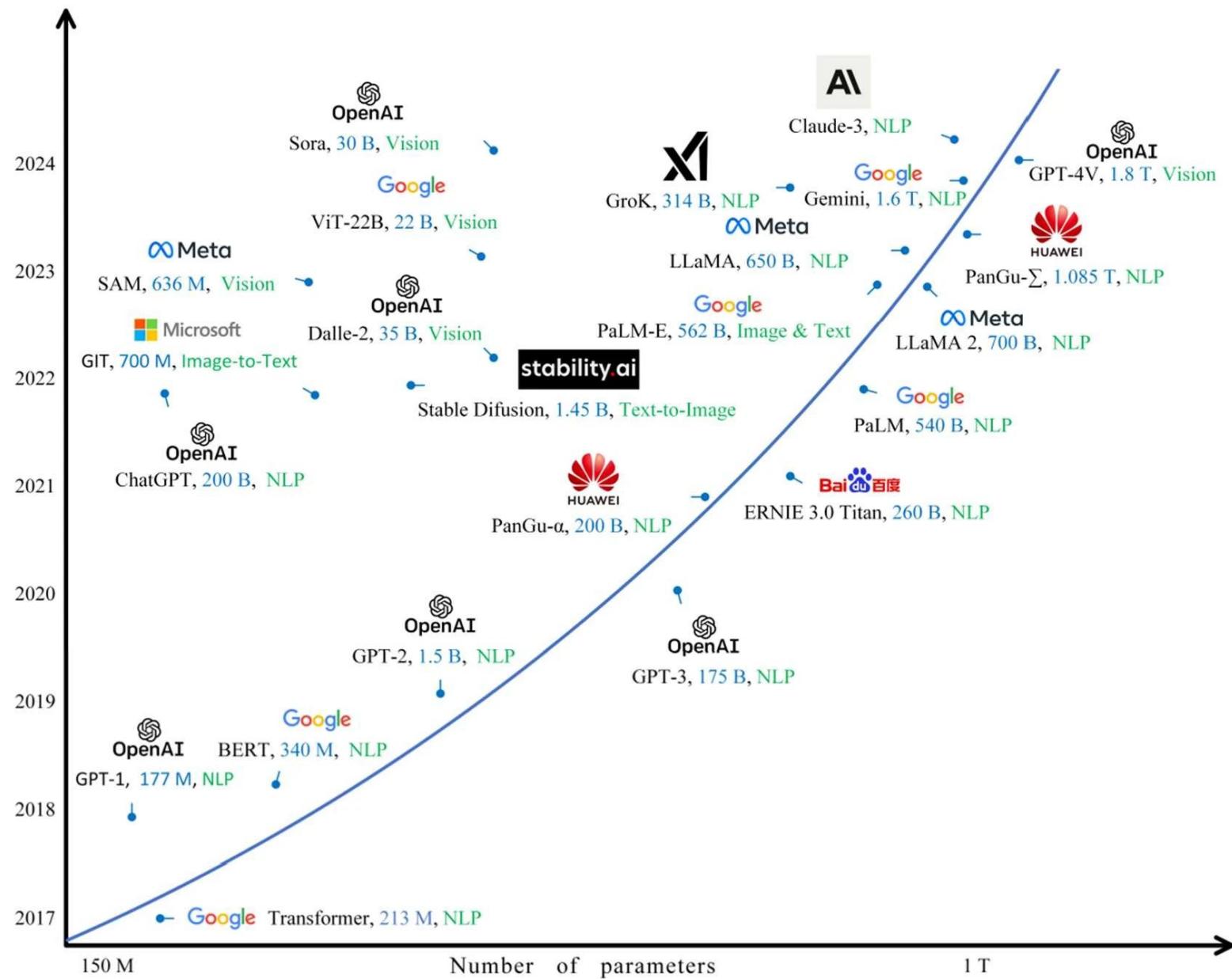


Mito 4:

Mientras más grande los LLMs
mejores los resultados.



tiempo vs # parámetros



Más parámetros no necesariamente implica mejor modelo

- Mayor capacidad del modelo se debería asociar directamente con un mejor modelo, pero hay problemas que afrontar
 - Capacidad de computo.
 - Datos de entrenamiento.
 - Tiempo de entrenamiento.
- Mientras la capacidad del modelo creció considerablemente desde GPT1.0 a GPT5.0, los nuevos modelos no muestran el avance esperado en comparación a su tamaño.
- Según Ilya Sutskever en NeurIPS 2024, los LLMs están llegando a un plateau en términos del proceso de aprendizaje. Es decir, hay que buscar una nueva forma de entrenamiento para seguir avanzando.

Mito 5: Los LLMs son inteligentes.

FALSO

Inteligencia versus conocimiento

- Normalmente asociamos inteligencia a conocimiento, pero no son lo mismo.
- Conocimiento => información acumulada.
- Inteligencia => saber usar la información.



Lector de libros

- Normalmente asociamos inteligencia a conocimiento, pero no son lo mismo.
- Los modelos actuales en vez de palabras usan “tokens”. Según los análisis, para GPT 5.0 se usaron alrededor de 70 trillones de tokens.
- 70 trillones de tokens es equivalente a 950 millones de libros.



Pero los LLM no se equivocan

- A pesar de tener mucho acceso a mucha información, los LLMs se equivocan más de lo que uno piensa.
- Ejemplo de errores en tareas cotidianas según chatGPT.

Área / Tarea	Error promedio
Conocimiento factual (datos, trivia, historia, ciencia)	10–30%
Razonamiento matemático (problemas paso a paso, álgebra, aritmética)	5–20%
Programación / Código	10–25%
Lenguaje natural (resúmenes, traducción, QA abierta)	5–15%

Pero los LLM no se equivocan

- A pesar de tener mucho acceso a mucha información, los LLMs se equivocan más de lo que uno piensa.
- Ejemplo de errores en tareas especializadas según chatGPT.

Tarea avanzada	Error promedio
Medicina (diagnóstico, interpretación clínica)	15–40%
Derecho (resolución de casos, exámenes de barra en EE.UU.)	15–35%
Matemáticas de investigación (demostraciones largas, álgebra abstracta)	40–70%
Programación compleja (proyectos multi-archivo, debugging avanzado)	20–50%
Ciencias duras (física cuántica, biología molecular)	25–50%

Conclusiones

Conclusiones y posibles soluciones

- Los LLMs son una ínfima parte de la IA.
- Los LLMs trabajan de forma muy distinta al cerebro humano, y al día de hoy no podrían llegar a ser la inteligencia artificial general.
- Los LLMs no son inteligentes, manejan mucha información, pero no piensan, por lo mismo se equivocan mucho más de lo que uno cree.
- Sin embargo hay varios tipos de soluciones para mejorar sus resultados:
 - Prompting techniques: hacer preguntas precisas y darle contexto al LLM.
 - Fine-tuning: se vuelve a entrenar el LLMs con datos particulares.
 - Retrieval-Augmented Generation: Combina el modelo con un buscador externo (vector DB, embeddings + indexación) para traer información contextual.

ICPRS 2025

- Quedan invitados al The IEEE 15th International Conference on Pattern Recognition Systems.
- 1 al 4 de Diciembre, UAI, Viña del Mar.



The banner features the UAI logo (Universidad Adolfo Ibáñez) and the text "IEEE ICPRS 2025". Below the text are three images illustrating pattern recognition:

- clock**: A flower bed shaped like a clock face with green numbers and a white airplane as the hour hand.
- building**: A modern building with palm trees in front, labeled with "building" and "palm".
- bridge**: A bridge over water at sunset, labeled with "bridge".

LAMDA-UAI

- 6 profesores colaboradores.
- 12 estudiantes (pregrado, magister, y doctorado).
- 13 graduados (pregrado y magister).



Muchas gracias

