

# Modelo de lenguaje grande

Un **modelo de lenguaje grande** o **LLM** (siglas en inglés para *Large Language Model*), también llamado **modelo de lenguaje de gran tamaño**, es un modelo de lenguaje que consta de una red neuronal con muchos parámetros (normalmente miles de millones o más), entrenados en grandes cantidades de texto sin etiquetar mediante aprendizaje autosupervisado o aprendizaje semisupervisado.<sup>1</sup> Los LLM surgieron alrededor de 2018 y se desempeñan bien en una amplia variedad de tareas. Esto ha cambiado el enfoque de la investigación del procesamiento del lenguaje natural alejándose del paradigma anterior de entrenar modelos supervisados especializados para tareas específicas.

Algunos LLM notables son la serie de modelos GPT de OpenAI (por ejemplo, GPT-3 y GPT-4 , utilizados en ChatGPT y Microsoft Copilot ), PaLM y Gemini de Google (el último de los cuales se utiliza actualmente en el chatbot del mismo nombre), o Claude de Anthropic, entre otros.

Aunque el término *modelo grande de lenguaje* no tiene una definición formal, a menudo se refiere a modelos de aprendizaje profundo que tienen un recuento de parámetros del orden de miles de millones o más. Los LLM son modelos de propósito general que se destacan en una amplia gama de tareas, en lugar de estar capacitados para una tarea específica (como el análisis de sentimientos, el reconocimiento de entidades nombradas o el razonamiento matemático).<sup>2</sup> La habilidad con la que realizan las tareas y la gama de tareas de las que son capaces parece ser una función de la cantidad de recursos (datos, número de parámetros, capacidad de cálculo) que se les dedican, de una manera que no depende sobre avances adicionales en el diseño.<sup>3</sup>

Aunque entrenados en tareas simples como predecir la siguiente palabra en una oración, se encuentran modelos de lenguaje neuronal con suficiente entrenamiento y conteo de parámetros para capturar gran parte de la sintaxis y la semántica del lenguaje humano. Además, los modelos de lenguaje grande demuestran un conocimiento general considerable sobre el mundo y son capaces de "memorizar" una gran cantidad de hechos durante el entrenamiento.

## Propiedades

### Conjuntos de datos de pre-entrenamiento

Los LLM están pre-entrenados en grandes conjuntos de corpus textuales. Algunos conjuntos de corpus de texto de uso común son Common Crawl, The Pile, MassiveText,<sup>4</sup> Wikipedia y GitHub. Los conjuntos de datos tienen un tamaño de hasta 10 billones de palabras.

El almacén de datos lingüísticos de alta calidad está dentro de los 4,6 a 17 billones de palabras, que está dentro de un orden de magnitud para los conjuntos de corpus textuales más grandes.<sup>5</sup>

### Leyes de escala

En general, un LLM se puede describir de manera incompleta con cuatro parámetros:<sup>[*cita requerida*]</sup> tamaño del modelo, tamaño del conjunto de datos de entrenamiento, costo del entrenamiento y rendimiento después del entrenamiento. Cada una de estas cuatro variables se puede definir con precisión mediante un número real y, empíricamente, se encuentra que dichos números están relacionados por leyes estadísticas simples, llamadas "leyes de escala".

Una ley de escala particular ("escala Chinchilla") para LLM entrenados autorregresivamente para una época, con un programa de tasa de aprendizaje logarítmico, establece que:<sup>6</sup>

$$\left\{ \begin{array}{l} C = C_0ND \\ L = \frac{A}{N^\alpha} + \frac{B}{D^\beta} + L_0 \end{array} \right.$$

donde las variables son:

$C$  es el costo de entrenar el modelo, en FLOPs.

$N$  es el número de parámetros en el modelo.

$D$  es el número de tokens en el conjunto de entrenamiento.

$L$  es la pérdida promedio de probabilidad logarítmica negativa por token (nats /token), lograda por el LLM capacitado en el conjunto de datos de prueba.

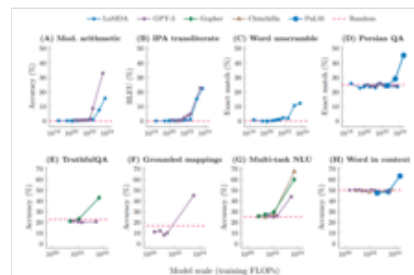
y los parámetros estadísticos son:

$C_0 = 6$ , lo que significa que cuesta 6 FLOP por parámetro entrenar en un token. Obsérvese que el costo de capacitación es mucho más alto que el costo de inferencia, donde cuesta de 1 a 2 FLOP por parámetro inferir en un token.

$\alpha = 0.34$ ,  $\beta = 0.28$ ,  $A = 406.4$ ,  $B = 410.7$ ,  $L_0 = 1.69$ .

## Habilidades emergentes

Si bien generalmente se puede extrapolar el rendimiento de los modelos grandes en varias tareas, en función del rendimiento de modelos más pequeños similares, a veces los modelos grandes experimentan un "cambio de fase discontinuo" en el que el modelo adquiere repentinamente habilidades sustanciales que no se ven en modelos más pequeños. Estas se conocen como "habilidades emergentes" y han sido objeto de un estudio sustancial. Los investigadores señalan que tales habilidades "no se pueden predecir simplemente extrapolando el rendimiento de modelos más pequeños".<sup>2</sup> Estas habilidades se descubren en lugar de programarse o diseñarse, en algunos casos solo después de que el LLM se haya implementado públicamente.<sup>3</sup> Se han descrito cientos de habilidades emergentes. Los ejemplos incluyen aritmética de varios pasos, aprobar exámenes de nivel universitario, identificar el significado previsto de una palabra,<sup>2</sup> indicaciones de cadena de pensamientos,<sup>2</sup> decodificar el Alfabeto Fonético Internacional, descifrar las letras de una palabra, identificar contenido ofensivo en párrafos de Hinglish (una combinación de hindi e inglés), y generando un equivalente en inglés similar a los proverbios en kiswahili.<sup>7</sup>



En una serie de puntos de referencia del lenguaje natural que involucran tareas como responder preguntas, los modelos no se desempeñan mejor que el azar hasta que alcanzan una cierta escala (en este caso, medida por cómputo de entrenamiento), momento en el cual su rendimiento aumenta considerablemente. Estos son ejemplos de habilidades emergentes.

## Alucinación

Se ha observado que los LLM generativos afirman con confianza afirmaciones de hecho que no parecen estar justificadas por sus datos de entrenamiento, un fenómeno que se ha denominado "alucinación".<sup>8</sup>

## Arquitectura

Los modelos de lenguajes grandes han utilizado con mayor frecuencia la arquitectura de transformadores, que, desde 2018, se ha convertido en la técnica de aprendizaje profundo estándar para datos secuenciales (anteriormente, las arquitecturas recurrentes como LSTM eran las más comunes).

## Tokenización

Formalmente, los LLM son funciones matemáticas cuya entrada y salida son listas de números. En consecuencia, las palabras deben convertirse en números. En general, un LLM usa un tokenizador separado. Un tokenizador es una función biyectiva que mapea entre textos y listas de enteros. El tokenizador generalmente se adapta primero a todo el conjunto de datos de entrenamiento y luego *se congela* antes de que se entrene el LLM. Una opción común es la codificación de pares de bytes.

Otra función de los tokenizadores es la compresión de texto, que ahorra cómputo. Las palabras o frases comunes como "dónde está" se pueden codificar en un token, en lugar de 10 caracteres. La serie OpenAI GPT utiliza un tokenizador donde 1 token se asigna a alrededor de 4 caracteres, o alrededor de 0,75 palabras, en texto común en inglés.<sup>9</sup> El texto en inglés poco común es menos predecible, por lo tanto, menos comprimible, por lo que requiere más tokens para codificar.

Un tokenizador no puede generar enteros arbitrarios. Por lo general, solo generan números enteros en el rango  $\{0, 1, 2, \dots, V - 1\}$ , donde  $V$  se llama su tamaño de vocabulario.

Algunos tokenizadores son capaces de manejar texto arbitrario (generalmente al operar directamente en Unicode), pero otros no. Al encontrar texto no codificable, un tokenizador generaría un token especial (a menudo `o`) que representa "texto desconocido". Esto a menudo se escribe como `[UNK]`, como en el documento BERT.

Otro token especial que se usa comúnmente es `[PAD]` (a menudo `1`), para "relleno". Esto se usa porque los LLM generalmente se usan en lotes de texto a la vez, y estos textos no se codifican con la misma longitud. Dado que los LLM generalmente requieren que la entrada sea una matriz no irregular, los textos codificados más cortos deben rellenarse hasta que coincidan con la longitud del más largo.

## Output

El output de un LLM es una distribución de probabilidad sobre su vocabulario. Esto generalmente se implementa de la siguiente manera:

- Al recibir un texto, la mayor parte del LLM genera un vector  $y \in \mathbb{R}^V$  dónde  $V$  es su tamaño de vocabulario (definido arriba).
- el vector  $y$  se pasa a través de una función softmax para obtener  $\text{softmax}(y)$ .

En el proceso, el vector  $y$  generalmente se llama el vector logit no normalizado, y el vector  $\text{softmax}(y)$  se llama vector de probabilidad. Dado que el vector  $\text{softmax}(y)$  tiene  $V$  entradas, todas no negativas, y suman 1, podemos interpretarlo como una distribución de probabilidad sobre  $\{0, 1, 2, \dots, V - 1\}$  —es decir, es una distribución de probabilidad sobre el vocabulario del LLM.

Considerar que la función softmax se define matemáticamente sin parámetros para variar. En consecuencia, no está entrenada.

## Entrenamiento

La mayoría de los LLM se entrenan mediante preentrenamiento generativo, es decir, dado un conjunto de datos de entrenamiento de tokens de texto, el modelo predice los tokens en el conjunto de datos. Hay dos estilos generales de preentrenamiento generativo:<sup>10</sup>

- autorregresivo (estilo GPT, "predecir la siguiente palabra"): Dado un segmento de texto como "Me gusta comer", el modelo predice los siguientes tokens, como "helado".
- enmascarado ("estilo BERT",<sup>11</sup> "prueba cloze"): dado un segmento de texto como "Me gusta [MASCARILLA] [MASCARILLA] crema", el modelo predice los tokens enmascarados, como "comer helado".

Los LLM pueden recibir capacitación en tareas auxiliares que prueban su comprensión de la distribución de datos, como la predicción de la siguiente oración (NSP), en la que se presentan pares de oraciones y el modelo debe predecir si aparecen consecutivamente en el corpus de capacitación.<sup>11</sup>

Por lo general, los LLM están capacitados para minimizar una función de pérdida específica: la probabilidad logarítmica negativa promedio por token (también llamada pérdida de entropía cruzada).<sup>12</sup> Por ejemplo. si un modelo autorregresivo, dado "Me gusta comer", predice una distribución de probabilidad  $Pr(\cdot | \text{Me gusta comer})$  entonces la pérdida de probabilidad logarítmica negativa en este token es  $-\log Pr(\text{helado} | \text{Me gusta comer})$ .

Durante el entrenamiento, la pérdida de regularización también se utiliza para estabilizar el entrenamiento. Sin embargo, la pérdida de regularización generalmente no se usa durante las pruebas y la evaluación. También hay muchos más criterios de evaluación más allá de la probabilidad logarítmica negativa.

Los primeros LLM se formaron en un corpus lingüístico que tenía una dimensión de miles de millones de palabras.

GPT-1, el primer modelo de la serie numerada de modelos de transformadores preentrenados generativos de OpenAI, se entrenó en 2018 en BookCorpus, que consta de 985 millones de palabras.<sup>13</sup> En el mismo año, BERT se capacitó en una combinación de BookCorpus y Wikipedia en inglés, con un total de 3300 millones de palabras.<sup>11</sup> Desde entonces, los corpus de capacitación para LLM han aumentado en órdenes de magnitud, llegando a billones de tokens.<sup>11</sup>

## Costo de entrenamiento

Los LLM son computacionalmente costosos de entrenar. Un estudio de 2020 estimó el costo de entrenar un modelo de 1500 millones de parámetros (2 órdenes de magnitud más pequeño que el estado del arte en ese momento) en \$1,6 millones. Los avances en software y hardware han reducido sustancialmente el costo, con un documento de 2023 que informa un costo de 72,300 A100-GPU -horas para entrenar un modelo de 12 mil millones de parámetros.<sup>14</sup> Se estima que para entrenar solo una vez a GPT-3, con 175 mil millones de parámetros, se necesitan \$4,6 millones, para lo cual una sola RTX 8000, tardaría 665 años en terminar.<sup>15</sup>

Para los LLMs basados en transformers, cuesta 6 FLOP por parámetro entrenar en un token. Debe considerarse que el costo de capacitación es mucho más alto que el costo de inferencia, donde cuesta de 1 a 2 FLOP por parámetro inferir en un token.<sup>[*cita requerida*]</sup>

Respecto a su coste ambiental, entrenar un LLM tiene un coste energético muy alto. Esto llega a tal punto que entrenar un modelo tan solo una vez, genera las mismas emisiones de carbono que un pasajero en un vuelo de Nueva York a San Francisco.<sup>16</sup> En el caso particular de GPT-3, se estima que entrenarlo una vez llega a consumir más de 1.200 MWh, produciendo más de 500 toneladas de emisiones de CO2.<sup>17</sup> Estos costes aumentan a medida que los modelos tengan más parámetros, es decir, sean más grandes y complejos. Cada ciclo de entrenamiento requiere la dedicación exclusiva de cientos o miles de CPUs y GPUs, que soportan una gran carga computacional, además de almacenar y mover bastas cantidades de datos. Todo esto contribuye a un gran consumo energético, y genera grandes cantidades de calor.

Para reducir estos costes, una solución posible es utilizar modelos más pequeños. Estos pueden tener un desempeño similar a un modelo grande en la mayoría de las situaciones, costando alrededor de \$100 para entrenar, en vez de millones. Un ejemplo de un modelo pequeño es el modelo Alpaca, desarrollado por investigadores de la Universidad de Stanford a partir del modelo de Meta AI llamado LLaMA. Este es suficientemente ligero como para correr en un ordenador de escritorio.<sup>18</sup>

Otra forma de reducir el coste de entrenamiento inicial es usar aprendizaje one-shot o few-shot durante su ciclo de vida, lo cual permitirá al modelo aprender durante su utilización. De esta manera, no se requiere una inversión tan grande para el entrenamiento inicial, y puede aprender a medida que se utiliza.

## Aplicación a tareas posteriores (downstream tasks)

Entre 2018 y 2020, el método estándar para preparar un LLM para una tarea específica de procesamiento del lenguaje natural (NLP) fue ajustar el modelo con capacitación adicional específica para la tarea. Posteriormente, se descubrió que los LLM más potentes, como GPT-3, pueden resolver tareas sin capacitación adicional a través de técnicas de "incitación", en las que el problema a resolver se presenta al modelo como un mensaje de texto, posiblemente con algunos ejemplos textuales de similares problemas y sus soluciones.

## Ajuste fino (Fine-tuning)

El ajuste fino es la práctica de modificar un modelo de lenguaje previamente entrenado entrenándolo (de manera supervisada) en una tarea específica (por ejemplo, análisis de sentimientos, reconocimiento de entidades nombradas o etiquetado de partes del discurso).<sup>19 20</sup> Es una forma de transferencia de aprendizaje. Por lo general, implica la introducción de un nuevo conjunto de pesos (weights)<sup>21</sup> que conectan la capa final del modelo de lenguaje con el resultado de la tarea posterior. Los pesos originales del modelo de lenguaje pueden "congelarse", de modo que solo se aprenda la nueva capa de pesos que los conecta con la salida durante el entrenamiento. Alternativamente, los pesos originales pueden recibir pequeñas actualizaciones (posiblemente con capas anteriores congeladas).<sup>11</sup>

## Indicaciones (prompting)

En el paradigma de indicaciones, popularizado por GPT-3,<sup>2</sup> el problema a resolver se formula a través de un mensaje de texto, que el modelo debe resolver proporcionando una finalización (a través de la inferencia). En las "indicaciones de pocas oportunidades", la indicación incluye una pequeña cantidad de ejemplos de pares similares (problema, solución). Por ejemplo, una tarea de análisis de opinión de etiquetar la opinión de una reseña de una película podría solicitarse de la siguiente manera:<sup>2</sup>

Reseña: Esta película apesta. Sentimiento: negativo
Reseña: ¡Esta película es fantástica! Sentimiento: positivo

Si el modelo da como resultado "positivo", entonces ha resuelto correctamente la tarea. En las indicaciones de disparo cero (zero-shot),<sup>Notas 1 22 23</sup> no se proporcionan ejemplos de resolución. Un ejemplo de un aviso de disparo cero para la misma tarea de análisis de sentimiento sería "El sentimiento asociado con la reseña de la película '¡Esta película es fantástica!' ".

Se ha demostrado que el rendimiento de pocos disparos de los LLM logra resultados competitivos en tareas de PNL, a veces superando los enfoques de ajuste fino de última generación. Ejemplos de tales tareas de PNL son la traducción, la respuesta a preguntas, las tareas de cloze<sup>Notas 2</sup>, descifrar palabras y usar una palabra nueva en una oración. La creación y optimización de dichos avisos se denomina ingeniería de avisos.

### Ajuste de instrucciones

El ajuste de instrucciones es una forma de ajuste fino diseñado para facilitar interacciones de indicaciones de tiro cero más naturales y precisas. Dada una entrada de texto, un modelo de lenguaje previamente entrenado generará una terminación que coincida con la distribución del texto en el que fue entrenado. Un modelo de lenguaje ingenuo dado el mensaje "Escribe un ensayo sobre los temas principales de *Hamlet* ". podría proporcionar una finalización como "Se aplicará una multa por retraso del 10% por día a las presentaciones recibidas después del 17 de marzo". En el ajuste de instrucciones, el modelo de lenguaje se entrena en muchos ejemplos de tareas formuladas como instrucciones en lenguaje natural, junto con las respuestas apropiadas.

En la práctica se han aplicado diversas técnicas para la puesta punto de instrucciones.<sup>23</sup> Un ejemplo, "autoinstrucción", ajusta el modelo de lenguaje en un conjunto de ejemplos de entrenamiento que son generados por un LLM (arrancado a partir de un pequeño conjunto inicial de ejemplos generados por humanos).

### Aprendizaje reforzado

El protocolo InstructGPT<sup>24</sup> de OpenAI implica un ajuste fino supervisado en un conjunto de datos de pares generados por humanos (solicitud, respuesta), seguido de un aprendizaje reforzado a partir de la retroalimentación humana (RLHF),<sup>25</sup> en el que se supervisó y aprendió un modelo de recompensa en un conjunto de datos de preferencias humanas, luego este modelo de recompensa se utilizó para capacitar al propio LLM mediante la optimización de políticas proximales.

## Evaluación

### Perplejidad

La medida más utilizada del rendimiento de un modelo de lenguaje es su perplejidad en un corpus de texto dado. La perplejidad es una medida del acierto con el que un modelo puede predecir el contenido de un conjunto de datos; cuanto mayor sea la probabilidad que el modelo asigna al conjunto de datos, menor será la perplejidad. Matemáticamente, la perplejidad se define como el exponencial de la probabilidad logarítmica negativa promedio por token:

$$\log(\text{Perplexity}) = -\frac{1}{N} \sum_{i=1}^N \log(\textit{Pr}(\text{token}_i | \text{context for token}_i))$$

aquí *N* es el número de tokens en el corpus de texto, y el "contexto para el token i" depende del tipo específico de LLM utilizado. Si el LLM es autorregresivo, entonces el "contexto para el token i" es el segmento de texto que aparece antes

del token *i*. Si el LLM está enmascarado, entonces el "contexto para el token *i*" es el segmento de texto que rodea al token *i*.<sup>26</sup>

Debido a que los modelos de lenguaje pueden sobreaajustarse a sus datos de entrenamiento, los modelos generalmente se evalúan por su perplejidad en un conjunto de prueba de datos no vistos.<sup>11</sup> Esto presenta desafíos particulares para la evaluación de grandes modelos de lenguaje. A medida que se entrenan en corpus de texto cada vez más grandes extraídos en gran parte de la web, es cada vez más probable que los datos de entrenamiento de los modelos incluyan inadvertidamente partes de cualquier conjunto de prueba dado.<sup>23</sup>

## Conjuntos de datos y puntos de referencia específicos de la tarea

También se ha desarrollado una gran cantidad de conjuntos de datos de prueba y puntos de referencia para evaluar las capacidades de los modelos de lenguaje en tareas posteriores más específicas. Las pruebas pueden diseñarse para evaluar una variedad de capacidades, incluido el conocimiento general, el razonamiento de sentido común y la resolución de problemas matemáticos.

Una amplia categoría de conjuntos de datos de evaluación son los conjuntos de datos de preguntas y respuestas, que consisten en pares de preguntas y respuestas correctas, por ejemplo, ("¿Han ganado los San Jose Sharks la Copa Stanley?"). , "No").<sup>27</sup> Una tarea de respuesta a una pregunta se considera un "libro abierto" si el mensaje del modelo incluye texto del que se puede derivar la respuesta esperada (por ejemplo, la pregunta anterior podría ir acompañada de algún texto que incluya la oración "Los Shraaks han avanzado a la Copa Stanley finales una vez, perdiendo ante los Pittsburgh Penguins en 2016").<sup>27</sup> De lo contrario, la tarea se considera "libro cerrado" y el modelo debe basarse en el conocimiento retenido durante el entrenamiento.<sup>28</sup> Algunos ejemplos de conjuntos de datos de respuesta a preguntas de uso común incluyen TruthfulQA, Web Questions, TriviaQA y SQuAD.<sup>28</sup>

Los conjuntos de datos de evaluación también pueden tomar la forma de finalización de texto, haciendo que el modelo seleccione la palabra o la oración más probable para completar un mensaje, por ejemplo: "Alice era amiga de Bob. Alice fue a visitar a su amiga, \_\_\_\_".<sup>23</sup>

También se han desarrollado algunos puntos de referencia compuestos que combinan una diversidad de diferentes conjuntos de datos y tareas de evaluación. Los ejemplos incluyen GLUE, SuperGLUE, MMLU, BIG-bench y HELM.<sup>29</sup>  
<sup>28</sup>

Anteriormente, era estándar informar los resultados en una parte retenida de un conjunto de datos de evaluación después de realizar un ajuste fino supervisado en el resto. Ahora es más común evaluar un modelo previamente entrenado directamente a través de técnicas de indicación, aunque los investigadores varían en los detalles de cómo formulan las indicaciones para tareas particulares, particularmente con respecto a cuántos ejemplos de tareas resueltas se adjuntan a la indicación (es decir, el valor de *n* en la solicitud *de n* disparos).

## Evaluaciones construidas adversarialmente

Debido al rápido ritmo de mejora de los grandes modelos de lenguaje, los puntos de referencia de evaluación han sufrido una vida útil corta, con modelos de última generación que "saturan" rápidamente los puntos de referencia existentes, superando el rendimiento de los anotadores humanos, lo que lleva a esfuerzos para reemplazar o aumentar el punto de referencia con tareas más desafiantes.<sup>30</sup>

Algunos conjuntos de datos se han construido de manera contradictoria, centrándose en problemas particulares en los que los modelos de lenguaje existentes parecen tener un rendimiento inusualmente bajo en comparación con los humanos. Un ejemplo es el conjunto de datos TruthfulQA, un conjunto de datos de respuesta a preguntas que consta de 817 preguntas cuyos modelos de lenguaje son susceptibles de responder incorrectamente al imitar falsedades a las que fueron expuestos repetidamente durante el entrenamiento. Por ejemplo, un LLM puede responder "No" a la pregunta "¿Puedes enseñarle trucos nuevos a un perro viejo?" Debido a su exposición al idioma inglés , *no puedes enseñarle nuevos trucos a un perro viejo*,<sup>31</sup> aunque esto no es literalmente cierto.<sup>32</sup>

Otro ejemplo de un conjunto de datos de evaluación contradictorio es Swag y su sucesor, HellaSwag, colecciones de problemas en los que se debe seleccionar una de múltiples opciones para completar un pasaje de texto. Las finalizaciones incorrectas se generaron mediante el muestreo de un modelo de lenguaje y el filtrado con un conjunto de

clasificadores. Los problemas resultantes son triviales para los humanos, pero en el momento en que se crearon los conjuntos de datos, los modelos de lenguaje de última generación tenían poca precisión. Por ejemplo:

Vemos un cartel de un gimnasio. Luego vemos a un hombre hablando a la cámara y sentado y acostado sobre una pelota de ejercicios. El hombre... a) demuestra cómo aumentar el trabajo de ejercicio eficiente corriendo pelotas hacia arriba y hacia abajo. b) mueve todos sus brazos y piernas y desarrolla mucho músculo. c) luego toca la pelota y vemos una demostración de gráficos y poda de setos. d) realiza abdominales mientras tiene la pelota y habla.<sup>33</sup>










BERT selecciona b) como la finalización más probable, aunque la respuesta correcta es d).<sup>33</sup>

# Lista de modelos grandes de lenguaje

Lista de modelos grandes de lenguaje

Nombre	Fecha <sup>Notas 3</sup>	Empresa	Número de parámetros <sup>Notas 4</sup>	Tamaño	Licencia <sup>Notas 5</sup>	Notas
<a href="#">BERT</a>	2018	<a href="#">Google</a>	340 millones <sup>34</sup>	3.3 miles de millones de palabras <sup>34</sup>	✓ <b>Apache 2.0</b> <sup>35</sup>	Un modelo de lenguaje temprano e influyente, pero solo codificador y, por lo tanto, no está diseñado para ser impulsado (prompted) o generativo. <sup>36</sup>
XLNet	2019	<a href="#">Google</a>	~340 millones <sup>37</sup>	33 miles de millones de palabras		Una alternativa a BERT, diseñado solo como codificador <sup>38 39</sup>
<a href="#">GPT-2</a>	2019	<a href="#">OpenAI</a>	1.5 miles de millones	40GB <sup>40</sup> (~10 miles de millones tokens) <sup>41</sup>	✓ <b>MIT</b> <sup>42</sup>	Modelo de propósito general basado en la arquitectura del transformer
<a href="#">GPT-3</a>	2020	OpenAI	175 miles de millones <sup>43</sup>	499 miles de millones tokens <sup>41</sup>	<b>API web pública</b>	Una variante mejorada de GPT-3, denominada GPT-3.5, se puso a disposición del público a través de una interfaz web llamada ChatGPT en 2022. <sup>44</sup>
GPT-Neo	Marzo de 2021	<a href="#">EleutherAI</a>	2.7 miles de millones <sup>45</sup>	825 GiB <sup>46</sup>	✓ <b>MIT</b> <sup>47</sup>	La primera de una serie de alternativas gratuitas de GPT-3 lanzadas por EleutherAI. GPT-Neo superó a un modelo GPT-3 de tamaño equivalente en algunos puntos de referencia, pero fue significativamente peor que el GPT-3 más grande. <sup>47</sup>
GPT-J	Junio de 2021	EleutherAI	6 miles de millones <sup>48</sup>	825 GiB <sup>49</sup>	✓ <b>Apache 2.0</b>	Modelo de lenguaje de estilo GPT-3
Megatron-Turing NLG	Octubre de 2021 <sup>50</sup>	<a href="#">Microsoft</a> y <a href="#">Nvidia</a>	530 miles de millones <sup>51</sup>	338.6 miles de millones tokens <sup>51</sup>	✗ <b>Acceso web restringido</b>	Arquitectura estándar pero entrenada en un clúster de supercomputación.
Ernie 3.0 Titan	Diciembre de 2021	<a href="#">Baidu</a>	260 miles de millones <sup>52</sup>	4 Tb	✗ <b>Patentado</b>	LLM de idioma chino. Ernie Bot se basa en este modelo.
<a href="#">Claude</a> <sup>53</sup>	Diciembre de 2021	Anthropic	52 miles de millones <sup>54</sup>	400 miles de millones tokens <sup>54</sup>	<b>Beta cerrada</b>	Ajustado para el comportamiento deseable en las conversaciones. <sup>55</sup>



Nombre	Fecha <sup>Notas 3</sup>	Empresa	Número de parámetros <sup>Notas 4</sup>	Tamaño	Licencia <sup>Notas 5</sup>	Notas	
GLaM (Generalist Language Model)	Diciembre de 2021	Google	1.2 trillion <sup>56</sup>	1.6 trillion tokens <sup>56</sup>	 <b>Patentado</b>	Modelo reducido de expertos, lo que hace que sea más costoso entrenar pero más barato ejecutar inferencias en comparación con GPT-3.	
Gopher	Diciembre de 2021	<u>DeepMind</u>	280 miles de millones <sup>57</sup>	300 mil millones de tokens <sup>58</sup>	 <b>Patentado</b>		
LaMDA (Language Models for Dialog Applications)	Enero de 2022	Google	137 miles de millones <sup>59</sup>	1.56T de palabras, 168 miles de millones tokens <sup>60</sup>	 <b>Patentado</b>	Especializado para la generación de respuestas en conversaciones. Se utiliza en el chatbot Google Bard.	
GPT-NeoX	Febrero de 2022	EleutherAI	20 miles de millones <sup>61</sup>	825 GiB <sup>62</sup>	 <b>Apache 2.0</b>	Basado en la arquitectura Megatron	
<u>Chinchilla</u>	Marzo de 2022	<u>DeepMind</u>	70 miles de millones <sup>63</sup>	1.4 trillion tokens <sup>63</sup>	 <b>Patentado</b>	Modelo de parámetros reducidos entrenado con más datos. Usado en el bot Sparrow.	
PaLM (Pathways Language Model)	Abril de 2022	Google	540 miles de millones <sup>64</sup>	768 miles de millones tokens <sup>63</sup>	 <b>Patentado</b>	Destinado a alcanzar los límites prácticos de la escala del modelo	
OPT (Open Pretrained Transformer)	Mayo de 2022	<u>Meta</u>	175 miles de millones <sup>65</sup>	180 miles de millones tokens	<b>Investigación no comercial<sup>66</sup></b>	Arquitectura GPT-3 con algunas adaptaciones de Megatron	
YaLM 100B	Junio de 2022	<u>Yandex</u>	100 miles de millones <sup>67</sup>	1.7TB <sup>67</sup>	 <b>Apache 2.0</b>	Modelo inglés-ruso basado en Megatron-LM de Microsoft.	
Minerva	Junio de 2022	Google	540 miles de millones <sup>68</sup>	38.5B tokens de páginas web filtradas para contenido matemático y de artículos enviados al servidor de preimpresión arXiv <sup>68</sup>	 <b>Patentado</b>	LLM capacitado para resolver "cuestiones matemáticas y científicas utilizando el razonamiento paso a paso". <sup>69</sup> Minerva se basa en el modelo PaLM, más capacitado en datos matemáticos y científicos.	
BLOOM	Julio de 2022	Gran colaboración liderada por <u>Hugging Face</u>	175 miles de millones <sup>70</sup>	350 miles de millones tokens (1.6TB) <sup>71</sup>	 <b>responsable IA</b>	Esencialmente GPT-3 pero entrenado en un corpus multilingüe (30% inglés excluyendo lenguajes de programación)	
Galactica	Noviembre de 2022	<u>Meta</u>	120 miles de millones	106 miles de millones tokens <sup>72</sup>	<b>CC-BY-NC-4.0</b>	Entrenado en texto científico y modalidades.	
AlexaTM (Teacher	Noviembre de 2022	<u>Amazon</u>	20 billion <sup>73</sup>	1.3 trillion <sup>74</sup>	<b>API web pública<sup>75</sup></b>	arquitectura bidireccional de	

Nombre	Fecha <sup>Notas 3</sup>	Empresa	Número de parámetros <sup>Notas 4</sup>	Tamaño	Licencia <sup>Notas 5</sup>	Notas
Models)						secuencia a secuencia
LLaMA (Large Language Model Meta AI)	Febrero de 2023	Meta	65 miles de millones <sup>76</sup>	1.4 trillion <sup>76</sup>		Capacitado en un gran corpus de 20 idiomas para apuntar a un mejor rendimiento con menos parámetros. Investigadores de la Universidad de Stanford entrenaron un modelo afinado basado en pesos LLaMA, llamado Alpaca. <sup>77</sup>
<u>GPT-4</u>	Marzo de 2023	OpenAI	Número exacto desconocido, aproximadamente 1 billón <sup>Notas 6 78</sup>	Desconoci	API web pública	Disponible para usuarios de ChatGPT Plus y utilizado en varios productos.
Cerebras-GPT	Marzo de 2023	Cerebras	13 miles de millones <sup>79</sup>		✓ Apache 2.0	Entrenado con la fórmula de Chinchilla.
Falcon	Marzo de 2023	Technology Innovation Institute	40 miles de millones <sup>80</sup>	1 Trillion tokens (1TB) <sup>80</sup>	✗ Patentado	Se afirma que el modelo usa solo el 75% del cálculo de entrenamiento de GPT-3, el 40% de Chinchilla y el 80% de PaLM-62B.
BloombergGPT	Marzo de 2023	Bloomberg L.P.	50 miles de millones	363 miles de millones token conjunto de datos basado en las fuentes de datos de Bloomberg, plus 345 miles de millones tokens de conjuntos de datos de propósito general <sup>81</sup>	✗ Patentado	LLM entrenado en datos financieros de fuentes patentadas, que "supera a los modelos existentes en tareas financieras por márgenes significativos sin sacrificar el rendimiento en los puntos de referencia generales de LLM" <sup>81</sup>
<u>PanGu-Σ</u>	Marzo de 2023	Huawei	1.085 billones	329 miles de millones tokens <sup>82</sup>	✗ Patentado	
<u>OpenAssistant</u> <sup>83</sup>	Marzo de 2023	LAION	17 miles de millones	1.5 billones tokens	✓ Apache 2.0	Entrenado en datos abiertos de colaboración colectiva

## Véase también

- Aprendizaje autosupervisado
- Modelos fundacionales
- Riesgo existencial de la inteligencia artificial

## Referencias

1. Goled, Shraddha (7 de mayo de 2021). «Self-Supervised Learning Vs Semi-Supervised Learning: How They Differ» (https://analyticsindiamag.com/self-supervised-learning-vs-semi-supervised-learning-how-th