



Modelos de lenguaje y LLM: evolución, características y criterios de selección

Danny Murillo González

Investigador

Universidad Tecnológica de Panamá - CIDITIC

Universidad de Granada, España - Doctorando

CC BY-NC-SA 4.0

Atribución/Reconocimiento-NoComercial-Compartir Igual 4.0 Internacional

02/10/2026

Contenido

Introducción	3
Modelos de lenguaje y LLM.....	4
Definiciones	4
Diferencias principales.....	4
Términos y metodología.....	5
Las tres categorías presentadas en este documento	5
Cómo funciona un modelo de lenguaje	6
Entrenamiento e inferencia	7
Arquitectura, modalidad, capacidad y acceso	7
Características que describen a un modelo	7
Criterios de selección de los modelos incluidos	8
Evolución de modelo y LLM 2018–2026	9
Línea de tiempo de Modelos de Lenguaje y LLM	10
1. Modelos de lenguaje por año.....	10
2. Modelos de lenguaje por creador	10
3. Modelos de lenguaje por arquitectura / enfoque.....	10
4. Modelos de lenguaje por modalidad.....	10
5. Modelo de lenguaje por ventana de contexto.....	10
Modelo, aplicación, repositorio, API y plataforma	24
Ejecución mediante API frente a ejecución local	24
Hugging Face	25
Ejemplo de código en Phyton.....	25
Resultado	25
Ejecución local de modelos: Ollama y otros runtimes	26
OpenRouter.....	27
Cómo funciona	27
LLM Leaderboard de OpenRouter	28
Otros LeaderBoard	28
Características del catálogo de modelos en OpenRouter.....	30
Parámetros de generación de un LLM	31

Ejemplo: configuración reproducible en local y en OpenRouter	33
Qué evaluar para seleccionar un modelo	35
Criterios de selección	35
Ejemplos de selección de Modelo	36
Ejemplo 1. Clasificar 37.000 artículos científicos en una taxonomía temática.	36
Ejemplo 2. Encontrar trabajos similares en un repositorio de tesis no publicadas.	36
Decisión en el ejemplo 1.....	37
Decisión en el ejemplo 2.....	37
Referencias.....	38

Introducción

En pocos años, los modelos de lenguaje han pasado de ser herramientas de laboratorio a estar presentes en buscadores, asistentes y flujos de investigación. Muchas veces desconocemos cuál es el origen de estos o cual es la diferencia entre Modelos del Lenguaje y LLM.

Este documento tiene un doble propósito, el primero es ofrecer una definición de qué es un modelo de lenguaje, qué se entiende por LLM en el uso actual y cómo han evolucionado ambos entre 2018 y 2026, a través de 61 modelos seleccionados y clasificados según, creador, categoría, arquitectura o enfoque, modalidad y ventana de contexto. El segundo propósito es orientar al lector sobre qué características observar, cómo interpretarlas en una plataforma y qué evaluar para elegir un modelo adecuado para una actividad concreta.

La oferta de modelos crece cada mes y existen plataformas como OpenRouter que reúnen más de 400 modelos, y elegir entre todos ellos puede que no sea una tarea rutinaria, debido a que, no existe un mejor modelo, sino un modelo adecuado que cumpla la tarea con calidad suficiente, a un costo razonable y con las garantías de privacidad y reproducibilidad que exige cada proyecto.

Si bien existen otras plataformas parecidas a OpenRouter, he seleccionado esta por la experiencia al utilizar diversos modelos donde a través de ellos he evaluado sus resultados, ventana de contexto, costo y tiempo de respuesta. No es un documento técnico, sino más conceptual u orientativo, en vías de brindar un timeline de la evolución de los modelos y herramientas que facilitan su uso.

Modelos de lenguaje y LLM

Todo LLM (Large Language Model) es un modelo de lenguaje, pero no todo modelo de lenguaje se describe necesariamente como LLM. En esta guía, modelo de lenguaje se utiliza como término general para sistemas que aprenden regularidades y representaciones del lenguaje, LLM se reserva principalmente para modelos entrenados a gran escala que, en el uso moderno, suelen asociarse con generación, seguimiento de instrucciones y realización de múltiples tareas.

No existe una frontera universal basada en un número exacto de parámetros, una arquitectura única o una capacidad específica que determine cuándo un modelo pasa a ser un LLM. La literatura y la divulgación técnica emplean el término con distinta amplitud, por ello, algunas fuentes incluyen modelos como BERT dentro de los LLM, mientras otras reservan el término para modelos generativos modernos.

1. En el uso actual, los LLM más conocidos son **generativos**: producen texto, código u otras salidas de forma autoregresiva, aunque la generación no constituye por sí sola una definición universal de LLM.
2. La **escala** sigue siendo una característica central: estos modelos se entrenan con grandes volúmenes de datos y altos recursos de cómputo, pero tampoco existe un umbral de parámetros aceptado de forma universal.
3. Muchos LLM modernos son de **propósito general** y pueden cambiar de tarea mediante instrucciones, pero también existen modelos grandes especializados en ciencia, código, medicina u otros dominios.

Esta distinción tiene una consecuencia práctica, no siempre hace falta un LLM. Para buscar artículos similares en un repositorio, un modelo de embeddings como BGE o SPECTER puede ser más rápido y económico, para redactar, razonar o seguir instrucciones complejas, un LLM generativo suele ser más apropiado.

Definiciones

- **Modelo de lenguaje**: sistema que aprende regularidades estadísticas y representaciones del lenguaje a partir de datos, según su diseño, puede predecir tokens, representar textos como vectores, comprender relaciones contextuales, clasificar o generar nuevas secuencias.
- **LLM (Large Language Model, modelo de lenguaje grande)**: modelo de lenguaje entrenado a gran escala sobre grandes volúmenes de datos. Actualmente, el término suele asociarse especialmente con modelos generativos capaces de seguir instrucciones y realizar múltiples tareas, aunque la literatura no establece un umbral único de tamaño ni una única arquitectura.

Diferencias principales

1. **Lo que producen**: un modelo de lenguaje puede devolver una etiqueta, un vector, una probabilidad o texto, según su arquitectura y entrenamiento, los LLM generativos modernos

producen texto libre, código o salidas estructuradas y pueden cambiar de tarea mediante instrucciones.

2. **La escala:** los modelos de lenguaje abarcan tamaños y objetivos muy diversos, los LLM se caracterizan por entrenarse a gran escala, pero el número de parámetros por sí solo no define de forma universal la categoría.
3. **El alcance:** algunos modelos están especializados en una tarea o representación concreta, mientras que muchos LLM modernos son multipropósito y pueden resumir, traducir, clasificar, programar o razonar a partir de instrucciones.

En esta guía, BERT, SBERT, BGE y SPECTER se agrupan como modelos de representación o embeddings; GPT-1, GPT-2, T5, BART y PEGASUS se presentan como modelos generativos precursores; y familias como GPT, Claude, Gemini, Llama, Qwen y DeepSeek se tratan como LLM generativos modernos. Esta agrupación es operativa y didáctica, no es una taxonomía universal.

Términos y metodología

Los términos sobre modelos de lenguaje y LLM no es completamente uniforme entre artículos académicos, documentación técnica y textos divulgativos. En este documento se adopta una clasificación operativa para facilitar la comprensión histórica y la selección práctica de modelos, es por eso por lo que las categorías utilizadas no pretenden establecer una frontera universal, sino organizar modelos con funciones y formas de uso diferentes bajo criterios explícitos.

Cuando una fuente utiliza el término LLM con un alcance más amplio, por ejemplo, incluyendo BERT, ello no contradice necesariamente esta guía, responde a un acuerdo terminológico distinto.

Las tres categorías presentadas en este documento

Los 61 modelos de las líneas de tiempo y de los anexos en este documento se han agrupado en tres categorías.

1. **Modelos de representación / embeddings:** comprenden el texto o lo convierten en vectores para clasificar, buscar o comparar. En este documento hay 17, entre ellos BERT, BETO, SciBERT, SBERT, SPECTER2 y BGE.
2. **Modelos generativos precursores:** generan texto y constituyen antecedentes directos de los LLM generativos modernos, suelen estar asociados a menor escala, menor contexto o tareas más delimitadas. El documento incluye 7, como GPT-1, GPT-2, T5, BART y PEGASUS.
3. **LLM generativos modernos:** modelos entrenados a gran escala, generalmente capaces de seguir instrucciones y realizar múltiples tareas. En la selección de esta guía constituyen la categoría más numerosa, con 37 modelos, desde GPT-3 hasta familias actuales de GPT, Claude, Gemini, Llama, Qwen y DeepSeek.

Un **modelo de representación** es muy útil hoy, ya que los sistemas de búsqueda semántica y RAG (*Retrieval-Augmented Generation*) suelen combinar un modelo de embeddings para encontrar documentos con un LLM para redactar la respuesta.

Cómo funciona un modelo de lenguaje

Aunque existen arquitecturas diferentes, el funcionamiento de muchos modelos puede entenderse como una secuencia de transformaciones.

El **texto de entrada** se divide primero en **tokens** (*unidades mínimas de texto que el modelo procesa, como una palabra, parte de una palabra, número o signo de puntuación*).

Cada token se convierte en una representación **numérica o embedding** (*un conjunto de números o vector que representa características del token o del texto*) y se combina con **información de posición** (*información numérica que indica el lugar que ocupa cada token dentro de la secuencia, permitiendo al modelo distinguir el orden de las palabras*), para conservar el orden de la secuencia.

Las capas **Transformer** (*arquitectura de red neuronal diseñada para procesar secuencias y relacionar sus elementos entre sí*) utilizan **mecanismos de atención** (*procedimientos que permiten al modelo determinar qué tokens son más relevantes entre sí dentro de una entrada*) para construir representaciones **dependientes del contexto** (*representaciones cuyo significado cambia según las palabras o tokens que rodean a cada elemento*).

En un **encoder** (*componente que procesa la entrada y genera una representación interna de su significado*), como BERT, estas representaciones se utilizan principalmente para comprender, clasificar o extraer información.

En un **decoder** (*componente que genera una salida de forma progresiva a partir de la información disponible*), como GPT, se utilizan para calcular qué token es más probable que aparezca a continuación. En un modelo **encoder-decoder** (*arquitectura que primero interpreta una secuencia de entrada y después genera otra a partir de esa representación*), como T5, una **secuencia de entrada** (*conjunto ordenado de tokens que recibe el modelo*) se transforma en una **secuencia de salida** (*conjunto ordenado de tokens generado como resultado, por ejemplo, una traducción, un resumen o una respuesta*).

En los modelos **generativos autoregresivos** (*modelos que generan el contenido un token cada vez, utilizando también los tokens generados anteriormente*), la generación se repite paso a paso.

El modelo estima una **distribución de probabilidades** (*conjunto de probabilidades asignadas a los posibles tokens que podrían aparecer a continuación*) y selecciona uno de acuerdo con parámetros de generación como la temperatura o el muestreo.

El token elegido se incorpora al **contexto** (*información disponible que el modelo puede considerar en ese momento, incluyendo instrucciones, texto de entrada y tokens ya generados*) y el proceso vuelve a repetirse para calcular el siguiente. En cambio, un **modelo de embeddings** (*modelo diseñado para transformar textos en vectores numéricos que representan su contenido semántico*) puede devolver directamente un **vector** (*lista ordenada de números que representa matemáticamente un texto*) para tareas de **similitud semántica** (*comparación de textos según su significado*), búsqueda, agrupamiento o clasificación.

Entrenamiento e inferencia

Es importante distinguir el entrenamiento de la inferencia. Durante el entrenamiento, el modelo aprende patrones a partir de datos y ajusta sus parámetros, durante la inferencia, esos parámetros ya aprendidos se utilizan para procesar una entrada y producir una salida.

Usar ChatGPT, ejecutar Llama en Ollama o llamar a un modelo mediante librerías o un API, son formas de inferencia, no de entrenamiento.

ASPECTO	ENTRENAMIENTO	INFERENCIA
OBJETIVO	Aprender patrones y ajustar parámetros	Utilizar los parámetros aprendidos
DATOS	Grandes corpus o conjuntos de entrenamiento	Prompt, documento o entrada del usuario
PARÁMETROS	Se modifican	Normalmente permanecen fijos
CUANDO OCURRE	Antes de distribuir o actualizar el modelo	Cada vez que se ejecuta una solicitud

Arquitectura, modalidad, capacidad y acceso

Estas dimensiones no deben tratarse como sinónimos, la arquitectura describe cómo está construido el modelo, la modalidad indica qué tipos de datos acepta o produce, la capacidad describe para qué tareas está preparado y el acceso indica cómo puede utilizarse. Separarlas evita confundir, por ejemplo, MoE con multimodalidad o un modelo abierto con una arquitectura concreta.

DIMENSIÓN	QUÉ DESCRIBE	EJEMPLOS
ARQUITECTURA / ENFOQUE	Cómo está construido o procesa la información	Encoder-only, decoder-only, encoder-decoder, MoE
MODALIDAD	Qué tipos de datos acepta o produce	Texto, imagen, audio, video
CAPACIDAD / ESPECIALIZACIÓN	Qué tarea o comportamiento prioriza	Embeddings, razonamiento, código, agentes
ACCESO	Cómo puede utilizarse	API, descargable, ejecución local, no publicado

Características que describen a un modelo

Estas son algunas de las características que conviene conocer al elegir un modelo, y que fueron utilizadas en las tablas y gráficos de Timeline de los modelos:

Característica	Explicación	Ejemplo
Tipo de arquitectura	Cómo está construido el modelo. <i>Encoder-only</i> comprende (BERT); <i>decoder-only</i> genera (GPT); <i>encoder-decoder</i> transforma un texto en otro (T5)	T5: encoder-decoder, transforma una secuencia de entrada en otra de salida, por ejemplo, para resumir o traducir.
Embeddings	Representación de un texto como un vector de números, textos parecidos quedan cerca	SPECTER convierte un resumen en un vector para buscar artículos similares
Mixture of Experts (MoE)	El modelo se divide en “expertos” y solo activa una parte en cada paso, lo que abarata el cálculo	Qwen3.8-Max tiene 2,4 billones de parámetros, pero activa 95.000 millones

Razonamiento	El modelo dedica tokens adicionales a pensar antes de responder, mejora problemas complejos y encarece la respuesta	OpenAI o1, DeepSeek-R1, Claude Opus 5.5
Modalidad	Tipos de entrada y salida que acepta: texto, imagen, audio, video, archivos	Gemini 3 acepta texto, imagen, audio y video
Token	Unidad mínima de texto que procesa el modelo, aproximadamente tres cuartos de una palabra en inglés	Clasificación documental ocupa unos 4–6 tokens
Ventana de contexto	Máximo de tokens que el modelo puede considerar a la vez, sumando instrucciones, documentos y respuesta	512 tokens en BERT frente a 1M en Claude Opus 5.5
Parámetros	Valores numéricos que el modelo ajusta al entrenar, indican su tamaño	BERT-base tiene 110 millones, los LLM actuales, cientos de miles de millones
Pesos	El archivo con los parámetros entrenados; es el modelo en sí	Los pesos de Llama 3 se pueden descargar, los de GPT-6, no
Acceso	Si el modelo se puede descargar y ejecutar en local o solo se usa en línea	Descargable (uso libre), Descargable (con licencia), Solo en línea, No publicado
Modelo <i>instruct</i>	Versión ajustada para seguir instrucciones y responder en el formato pedido	Mistral Small 3.2 Instruct, útil para clasificar y extraer datos
Cuantización	Reducción de la precisión numérica de los pesos para que el modelo ocupe menos y corra en equipos modestos	Un modelo en 4 bits ocupa cerca de la cuarta parte que en 16 bits

Criterios de selección de los modelos incluidos

La línea de tiempo no pretende constituir un inventario exhaustivo de todos los modelos publicados entre 2018 y 2026. Se seleccionaron modelos que representan al menos uno de los siguientes aspectos:

- Un avance en su arquitectura
- Una familia con impacto significativo en la evolución de los modelos de lenguaje
- Relevancia para embeddings, ciencia o recuperación de información
- Un cambio importante en ventana de contexto, modalidad, razonamiento o agentes
- Una familia de uso actual disponible mediante API o ejecución local.

Las especificaciones se refieren a las versiones indicadas y pueden variar entre checkpoints, submodelos, proveedores o APIs. Para valores como contexto, modalidad, parámetros y acceso se priorizan las fuentes oficiales y las model cards cuando están disponibles.

Evolución de modelo y LLM 2018–2026

En ocho años, la selección presentada en esta guía muestra el paso desde modelos centrados en representación y comprensión, con contextos de cientos de tokens, hasta LLM multimodales y de razonamiento que manejan ventanas de contexto de cientos de miles o millones de tokens.

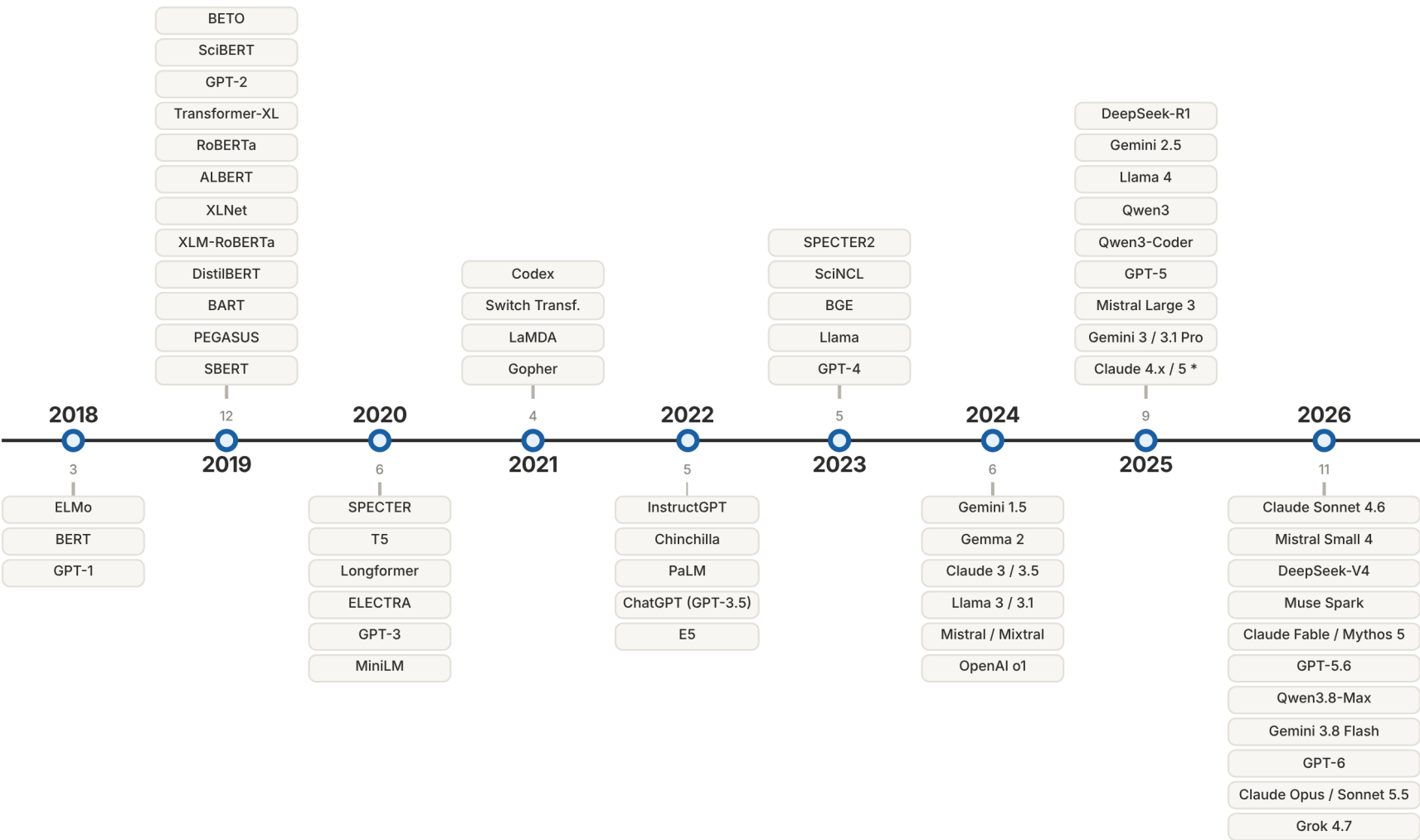
De los 61 modelos seleccionados, 37 se clasifican operativamente como LLM generativos modernos, todos los modelos seleccionados en esta guía a partir de 2024 pertenecen a esa categoría. Las cinco gráficas utilizan el mismo eje temporal, en las cuatro clasificaciones, los LLM aparecen sobre el eje y los demás modelos de lenguaje debajo, mientras que el borde punteado identifica a los modelos generativos precursores. La gráfica de arquitectura distingue encoder-only, encoder-decoder, decoder-only denso y Mixture of Experts (MoE). No detallado, no divulgado o no especificado, agrupa los modelos propietarios cuyo creador no ha publicado su arquitectura, modalidad o ventana de contexto.

Línea de tiempo de Modelos de Lenguaje y LLM

1. Modelos de lenguaje por año
2. Modelos de lenguaje por creador
3. Modelos de lenguaje por arquitectura / enfoque
4. Modelos de lenguaje por modalidad
5. Modelo de lenguaje por ventana de contexto

Modelos de lenguaje por año, 2018 – 2026

61 modelos seleccionados



Año de lanzamiento de cada modelo o familia seleccionado en esta guía. * Claude 4.x / 5 abarca 2025–2026.

Datos actualizados a octubre de 2026

Modelos de lenguaje por creador, 2018 – 2026

61 modelos seleccionados · 37 LLM

CREADOR

Google / DeepMind

OpenAI

Meta

Anthropic

Allen Institute for AI

Mistral AI

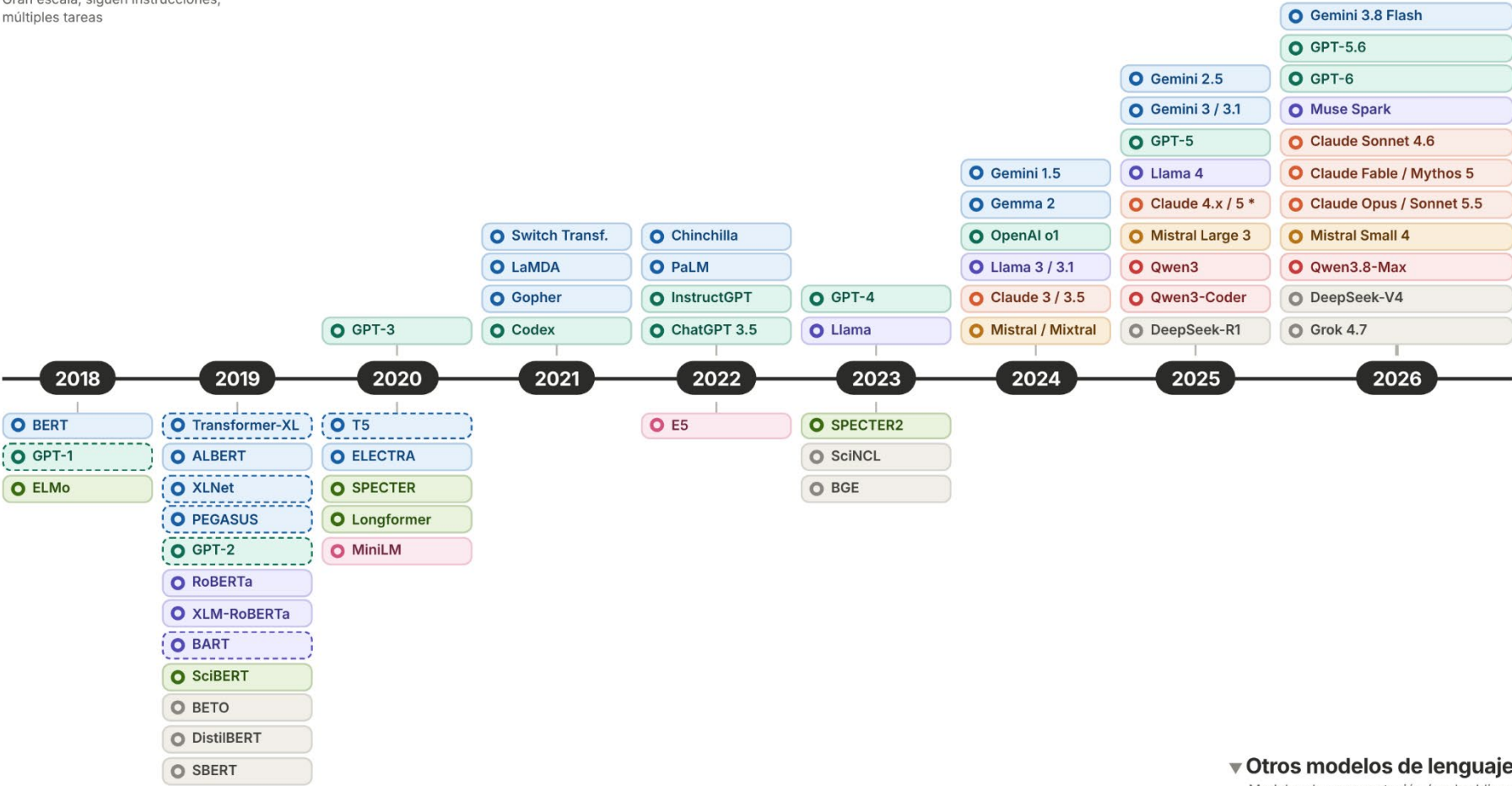
Microsoft Research

Alibaba / Qwen

Otros

▲ LLM generativos modernos

Gran escala, siguen instrucciones, múltiples tareas



▼ Otros modelos de lenguaje
Modelos de representación / embeddings
Modelos generativos precursores

Otros: Universidad de Chile, UKP Lab (TU Darmstadt), Hugging Face, investigación académica, BAAI, DeepSeek y xAI. * Claude 4.x / 5 abarca 2025–2026. Clasificación operativa y didáctica de esta guía. Datos a octubre de 2026

Modelos de lenguaje por arquitectura / enfoque, 2018 – 2026

61 modelos seleccionados · 37 LLM

ARQUITECTURA / ENFOQUE

Pre-Transformer (BiLSTM)

Encoder-only

Encoder-decoder

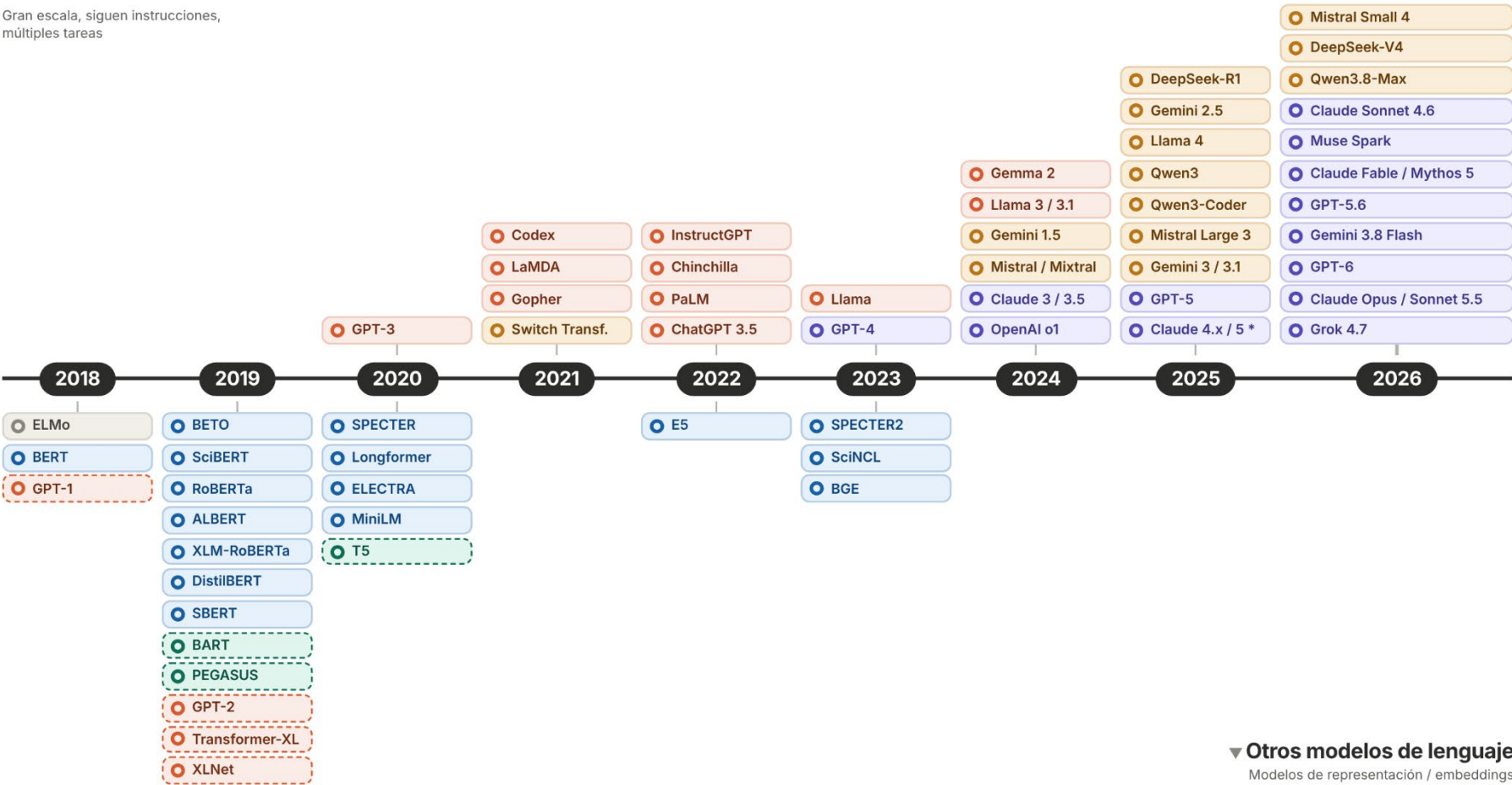
Decoder-only (denso)

Mixture of Experts (MoE)

No divulgada por el creador

▲ LLM generativos modernos

Gran escala, siguen instrucciones, múltiples tareas



▼ Otros modelos de lenguaje
Modelos de representación / embeddings
Modelos generativos precursores

«No divulgada»: modelo propietario sin informe técnico de arquitectura. * Claude 4.x / 5 abarca 2025–2026. Clasificación operativa y didáctica de esta guía. Datos a octubre de 2026

Modelos de lenguaje por modalidad, 2018 – 2026

61 modelos seleccionados · 37 LLM

MODALIDAD DE ENTRADA



Texto



Texto científico



Texto y código



Texto + imagen



Texto + imagen + video



Texto, imagen, audio y video

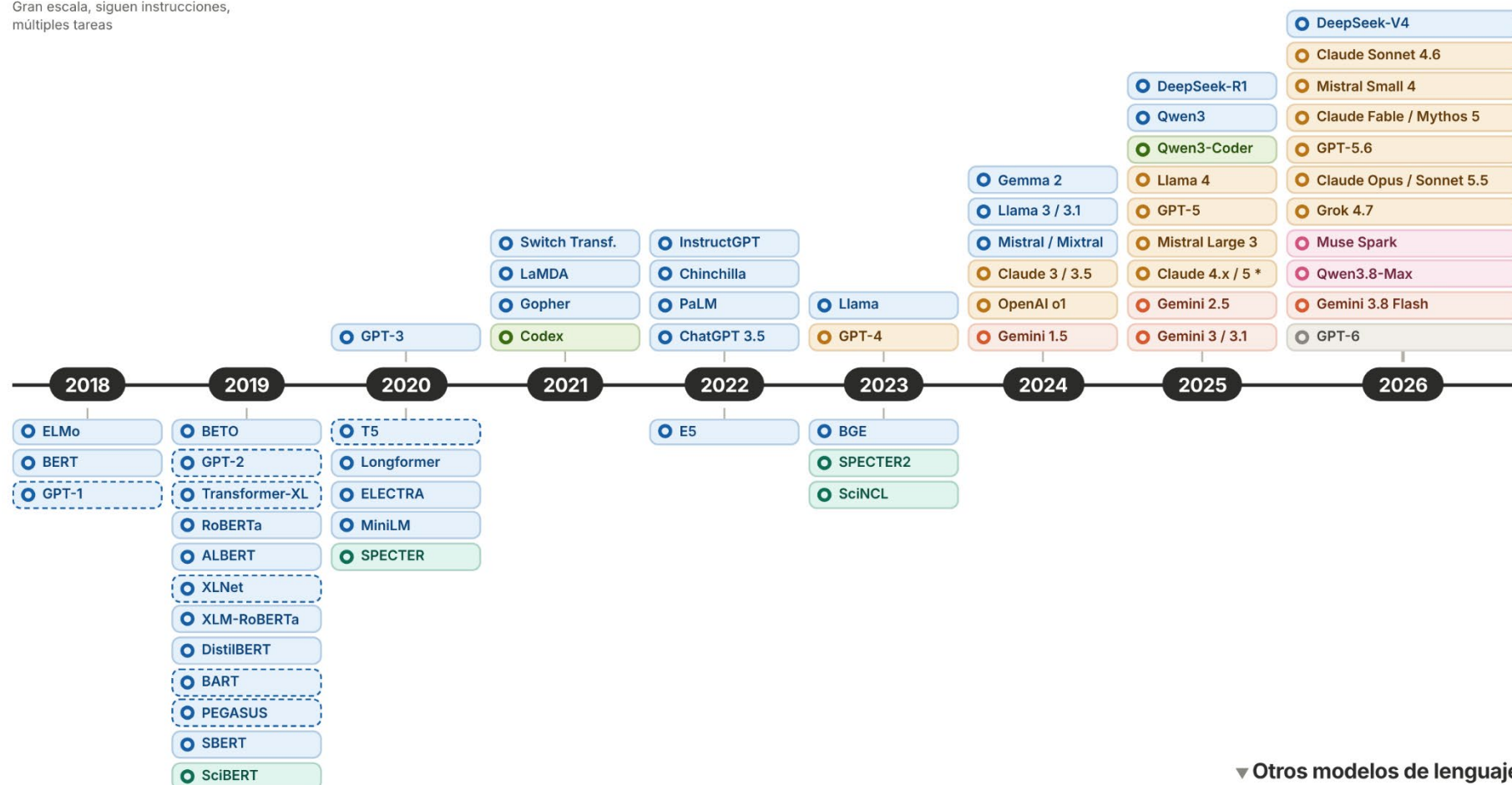


No detallado

▲ LLM generativos

modernos

Gran escala, siguen instrucciones, múltiples tareas



▼ Otros modelos de lenguaje

Modelos de representación / embeddings

Modelos generativos precursores

* Claude 4.x / 5 abarca 2025–2026. Clasificación operativa y didáctica de esta guía.

Datos a octubre de 2026

Modelos de lenguaje por ventana de contexto, 2018 – 2026

61 modelos seleccionados · 37 LLM

VENTANA DE CONTEXTO (VALOR MÁXIMO)

Hasta 1K

2K – 8K

16K – 128K

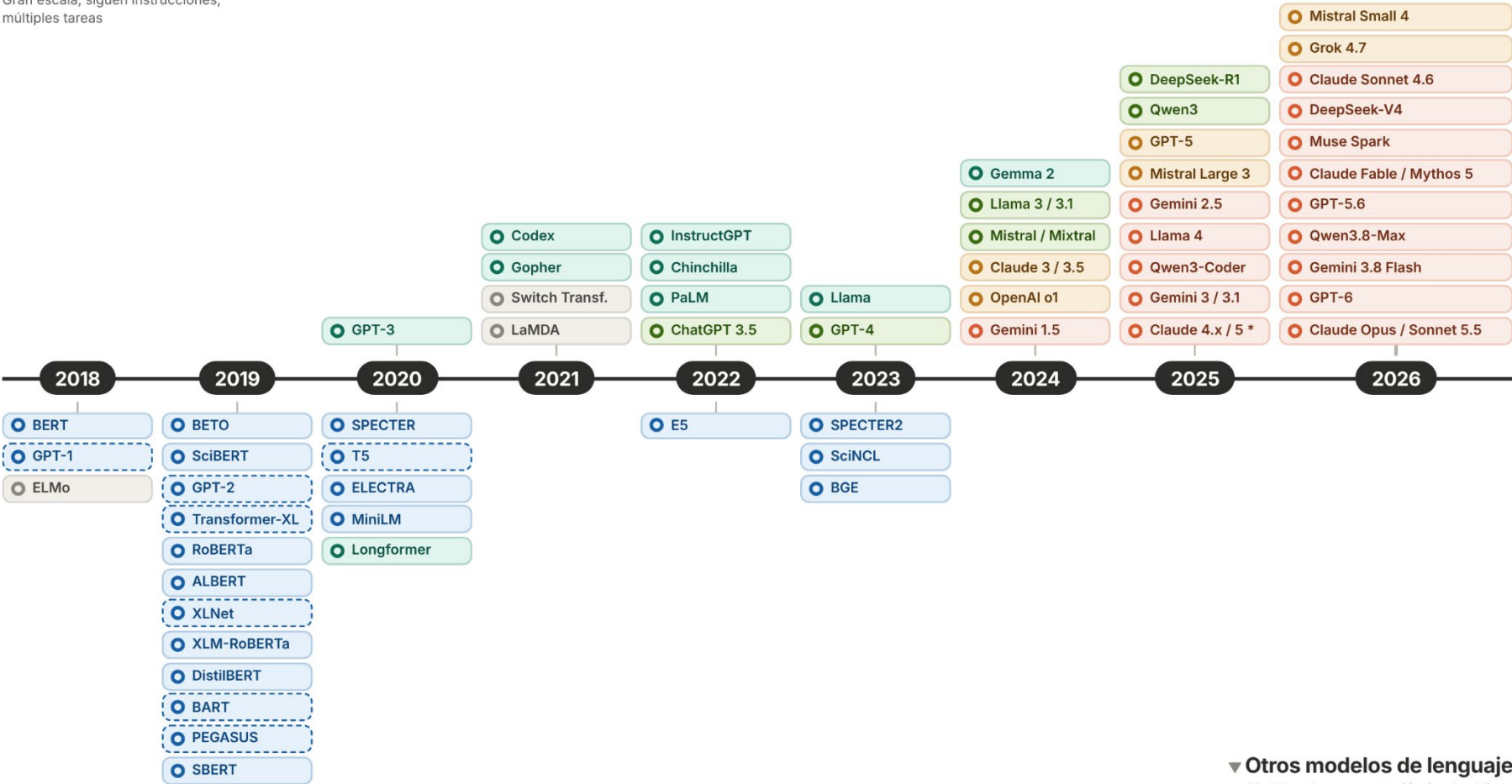
200K – 500K

1M o más

No especificado

▲ LLM generativos modernos

Gran escala, siguen instrucciones, múltiples tareas



▼ Otros modelos de lenguaje

Modelos de representación / embeddings

Modelos generativos precursores

El color agrupa por el valor máximo reportado; el valor exacto figura en la tabla. * Claude 4.x / 5 abarca 2025–2026. Clasificación operativa y didáctica de esta guía. Datos a octubre de 2026

Tabla 1. Modelos por año de creación, creador, categoría, arquitectura/enfoque, modalidad y ventana de contexto

Año	Modelo / familia	Categoría	Creador / organización	Arquitectura / enfoque	Modalidad	Ventana de contexto
2018	ELMo	Modelo de representación	Allen Institute for AI	Pre-Transformer (BiLSTM)	Texto	No aplica
2018	BERT	Modelo de representación	Google Research	Encoder-only	Texto	512 tokens
2018	GPT-1	Precursor generativo	OpenAI	Decoder-only (denso)	Texto	512 tokens
2019	BETO	Modelo de representación	Universidad de Chile / DCC	Encoder-only	Texto	512 tokens
2019	SciBERT	Modelo de representación	Allen Institute for AI	Encoder-only	Texto científico	512 tokens
2019	GPT-2	Precursor generativo	OpenAI	Decoder-only (denso)	Texto	1,024 tokens
2019	Transformer-XL	Precursor generativo	Google Brain / Carnegie Mellon University	Decoder-only (denso)	Texto	~512 por segmento + memoria recurrente
2019	RoBERTa	Modelo de representación	Meta AI / Facebook AI Research	Encoder-only	Texto	512 tokens
2019	ALBERT	Modelo de representación	Google Research / Toyota Technological Institute at Chicago	Encoder-only	Texto	512 tokens
2019	XLNet	Precursor generativo	Google Brain / Carnegie Mellon University	Decoder-only (denso)	Texto	512 tokens
2019	XLM-RoBERTa	Modelo de representación	Meta AI / Facebook AI Research	Encoder-only	Texto	512 tokens
2019	DistilBERT	Modelo de representación	Hugging Face	Encoder-only	Texto	512 tokens
2019	BART	Precursor generativo	Meta AI / Facebook AI Research	Encoder-decoder	Texto	1,024 tokens
2019	PEGASUS	Precursor generativo	Google Research	Encoder-decoder	Texto	1,024 tokens

2019	SBERT	Modelo de representación	UKP Lab, TU Darmstadt	Encoder-only	Texto	512 tokens
2020	SPECTER	Modelo de representación	Allen Institute for AI	Encoder-only	Texto científico	512 tokens
2020	T5	Precursor generativo	Google Research	Encoder-decoder	Texto	512 tokens
2020	Longformer	Modelo de representación	Allen Institute for AI	Encoder-only	Texto	4,096 tokens
2020	ELECTRA	Modelo de representación	Google Research	Encoder-only	Texto	512 tokens
2020	GPT-3	LLM	OpenAI	Decoder-only (denso)	Texto	2,048 tokens
2020	MiniLM	Modelo de representación	Microsoft Research	Encoder-only	Texto	512 tokens
2021	Codex	LLM	OpenAI	Decoder-only (denso)	Texto y código	4,096 tokens
2021	Switch Transformer	LLM	Google Research	Mixture of Experts (MoE)	Texto	Variable según variante
2021	LaMDA	LLM	Google	Decoder-only (denso)	Texto	No publicado
2021	Gopher	LLM	DeepMind	Decoder-only (denso)	Texto	2,048 tokens
2022	InstructGPT	LLM	OpenAI	Decoder-only (denso)	Texto	2,048 tokens
2022	Chinchilla	LLM	DeepMind	Decoder-only (denso)	Texto	2,048 tokens
2022	PaLM	LLM	Google Research	Decoder-only (denso)	Texto	2,048 tokens
2022	ChatGPT / GPT-3.5	LLM	OpenAI	Decoder-only (denso)	Texto	4K–16K tokens
2022	E5	Modelo de representación	Microsoft Research	Encoder-only	Texto	512 tokens
2023	SPECTER2	Modelo de representación	Allen Institute for AI	Encoder-only	Texto científico	512 tokens
2023	SciNCL	Modelo de representación	Investigación académica	Encoder-only	Texto científico	512 tokens
2023	BGE	Modelo de representación	BAAI	Encoder-only	Texto	512–1,024 tokens según variante

2023	Llama	LLM	Meta AI	Decoder-only (denso)	Texto	2,048 tokens
2023	GPT-4	LLM	OpenAI	No divulgada	Texto + imagen (entrada)	8K–32K tokens
2024	Gemini 1.5	LLM	Google DeepMind	Mixture of Experts (MoE)	Texto + imagen + audio + video	128K estándar; hasta 1M
2024	Gemma 2	LLM	Google DeepMind	Decoder-only (denso)	Texto	8K
2024	Claude 3 / 3.5	LLM	Anthropic	No divulgada	Texto + imagen (entrada)	200K
2024	Llama 3 / 3.1	LLM	Meta AI	Decoder-only (denso)	Texto	8K–128K
2024	Mistral / Mixtral	LLM	Mistral AI	Mixture of Experts (MoE)	Texto	32K
2024	OpenAI o1	LLM	OpenAI	No divulgada	Texto + imagen (entrada)	200K
2025	DeepSeek-R1	LLM	DeepSeek	Mixture of Experts (MoE)	Texto	128K
2025	Gemini 2.5	LLM	Google DeepMind	Mixture of Experts (MoE)	Texto + imagen + audio + video	1,048,576
2025	Llama 4	LLM	Meta AI	Mixture of Experts (MoE)	Texto + imagen	1M–10M según submodelo
2025	Qwen3	LLM	Alibaba Cloud / Qwen Team	Mixture of Experts (MoE)	Texto	32K–128K según variante
2025	Qwen3-Coder	LLM	Alibaba Cloud / Qwen Team	Mixture of Experts (MoE)	Texto y código	256K nativo; hasta 1M extendido
2025	GPT-5	LLM	OpenAI	No divulgada	Texto + imagen (entrada)	400K (API)
2025	Mistral Large 3	LLM	Mistral AI	Mixture of Experts (MoE)	Texto + imagen	256K
2025	Gemini 3 / 3.1 Pro	LLM	Google DeepMind	Mixture of Experts (MoE)	Texto + imagen + audio + video	1M
2025–2026	Claude 4.x / 5	LLM	Anthropic	No divulgada	Texto + imagen (entrada)	Hasta 1M
2026	Claude Sonnet 4.6	LLM	Anthropic	No divulgada	Texto + imagen (entrada)	1M (beta)
2026	Mistral Small 4	LLM	Mistral AI	Mixture of Experts (MoE)	Texto + imagen	256K

2026	DeepSeek-V4	LLM	DeepSeek	Mixture of Experts (MoE)	Texto	1M
2026	Muse Spark	LLM	Meta Superintelligence Labs	No divulgada	Texto + imagen + video + archivos	1M
2026	Claude Fable 5 / Mythos 5	LLM	Anthropic	No divulgada	Texto + imagen (entrada)	1M
2026	GPT-5.6 (Sol, Terra, Luna)	LLM	OpenAI	No divulgada	Texto + imagen (entrada)	1.05M
2026	Qwen3.8-Max	LLM	Alibaba Cloud / Qwen Team	Mixture of Experts (MoE)	Texto + imagen + video	1M
2026	Gemini 3.8 Flash	LLM	Google DeepMind	No divulgada	Texto + imagen + audio + video	~1M
2026	GPT-6 (Astra, Sol, Luna)	LLM	OpenAI	No divulgada	No detallado en las fuentes consultadas	1.05M
2026	Claude Opus 5.5 / Sonnet 5.5	LLM	Anthropic	No divulgada	Texto + imagen (entrada)	1M
2026	Grok 4.7	LLM	xAI	No divulgada	Texto + imagen (entrada)	500K

Categoría

- LLM generativo moderno: modelo de lenguaje entrenado a gran escala que, en esta guía, se caracteriza por generación, seguimiento de instrucciones y capacidades generales o amplias.
- Modelo de representación / embeddings: comprende texto o lo convierte en vectores para clasificar, buscar o comparar; su salida principal no es una respuesta generativa libre.
- Modelo generativo precursor: genera texto y constituye un antecedente directo de los LLM generativos modernos; se incluye como categoría histórica y operativa de esta guía.

Tabla 2. Modelos por año de creación, características y acceso

Año	Modelo / familia	Categoría	Característica principal	Acceso
2018	ELMo	Modelo de representación	Introdujo embeddings contextuales de palabras mediante redes BiLSTM, anticipando la representación contextual moderna.	Descargable (uso libre)
2018	BERT	Modelo de representación	Transformer bidireccional que revolucionó la comprensión del lenguaje mediante <i>Masked Language Modeling</i> .	Descargable (uso libre)
2018	GPT-1	Precursor generativo	Primer GPT; estableció el preentrenamiento generativo seguido de ajuste para tareas específicas.	Descargable (uso libre)
2019	BETO	Modelo de representación	Adaptó BERT específicamente al español mediante preentrenamiento sobre grandes corpus en este idioma.	Descargable (uso libre)
2019	SciBERT	Modelo de representación	Adaptó BERT al lenguaje científico mediante entrenamiento sobre artículos académicos de múltiples disciplinas.	Descargable (uso libre)
2019	GPT-2	Precursor generativo	Demostró que aumentar la escala de los Transformers generativos mejoraba las capacidades transferibles entre tareas.	Descargable (uso libre)
2019	Transformer-XL	Precursor generativo	Introdujo recurrencia entre segmentos para mantener contexto durante secuencias de texto más extensas.	Descargable (uso libre)
2019	RoBERTa	Modelo de representación	Optimizó el entrenamiento de BERT eliminando NSP y mejorando datos, hiperparámetros y enmascaramiento dinámico.	Descargable (uso libre)
2019	ALBERT	Modelo de representación	Redujo parámetros de BERT mediante compartición de parámetros y factorización de embeddings.	Descargable (uso libre)
2019	XLNet	Precursor generativo	Combinó aprendizaje autoregresivo permutado con Transformer-XL para capturar contexto bidireccional.	Descargable (uso libre)
2019	XLNet	Precursor generativo	Combinó aprendizaje autoregresivo permutado con Transformer-XL para capturar contexto bidireccional.	Descargable (uso libre)
2019	XLM-RoBERTa	Modelo de representación	Extendió RoBERTa al procesamiento multilingüe mediante entrenamiento sobre grandes corpus en numerosos idiomas.	Descargable (uso libre)
2019	DistilBERT	Modelo de representación	Aplicó destilación de conocimiento para obtener una versión de BERT más pequeña, rápida y eficiente.	Descargable (uso libre)
2019	BART	Precursor generativo	Combinó encoder bidireccional y decoder autoregresivo para generación, reconstrucción y resumen.	Descargable (uso libre)
2019	PEGASUS	Precursor generativo	Especializó el preentrenamiento Transformer en resumen automático mediante predicción de oraciones eliminadas.	Descargable (uso libre)
2019	SBERT	Modelo de representación	Adaptó BERT mediante redes siamesas para generar embeddings comparables de oraciones y textos.	Descargable (uso libre)
2020	SPECTER	Modelo de representación	Generó embeddings de artículos científicos utilizando texto y relaciones de citación para representar similitud temática.	Descargable (uso libre)

2020	T5	Precursor generativo	Unificó múltiples tareas de PLN bajo el paradigma <i>text-to-text</i> .	Descargable (uso libre)
2020	Longformer	Modelo de representación	Incorporó atención dispersa para procesar documentos mucho más largos que los Transformers convencionales.	Descargable (uso libre)
2020	ELECTRA	Modelo de representación	Sustituyó la predicción de tokens enmascarados por detección de tokens reemplazados para mejorar eficiencia.	Descargable (uso libre)
2020	GPT-3	LLM	Popularizó el <i>in-context learning</i> y el aprendizaje <i>zero-shot</i> y <i>few-shot</i> a gran escala.	Solo en línea
2020	MiniLM	Modelo de representación	Aplicó destilación de la autoatención para crear Transformers compactos, base de muchos modelos de similitud semántica.	Descargable (uso libre)
2021	Codex	LLM	Adaptó GPT al código fuente y mostró el potencial de los LLM especializados en programación.	Solo en línea
2021	Switch Transformer	LLM	Demostó la escalabilidad de arquitecturas <i>Mixture of Experts</i> activando solo una fracción del modelo.	Descargable (uso libre)
2021	LaMDA	LLM	Familia optimizada específicamente para mantener conversaciones abiertas y coherentes.	No publicado
2021	Gopher	LLM	Modelo de gran escala que ayudó a estudiar cómo el tamaño afecta conocimiento, razonamiento y sesgos.	No publicado
2022	InstructGPT	LLM	Popularizó RLHF para alinear modelos generativos con instrucciones y preferencias humanas.	Solo en línea
2022	Chinchilla	LLM	Mostró la importancia de equilibrar parámetros, datos y cómputo en las leyes de escalamiento.	No publicado
2022	PaLM	LLM	Modelo de gran escala que destacó por aprendizaje <i>few-shot</i> y uso del sistema Pathways.	Solo en línea
2022	ChatGPT / GPT-3.5	LLM	Popularizó mundialmente los LLM conversacionales capaces de seguir instrucciones y mantener contexto.	Solo en línea
2022	E5	Modelo de representación	Embeddings generales entrenados de forma contrastiva a gran escala, eficaces para búsqueda y similitud sin ajuste adicional.	Descargable (uso libre)
2023	SPECTER2	Modelo de representación	Amplió SPECTER con representaciones científicas más robustas y adaptables a múltiples tareas académicas.	Descargable (uso libre)
2023	SciNCL	Modelo de representación	Usó la red de citas para elegir ejemplos contrastivos y mejorar los embeddings de documentos científicos.	Descargable (uso libre)
2023	BGE	Modelo de representación	Familia de embeddings multilingües optimizada para recuperación y ranking, muy usada como base de sistemas RAG.	Descargable (uso libre)
2023	Llama	LLM	Impulsó el ecosistema de modelos con pesos disponibles para investigación, adaptación y ejecución local.	Descargable (con licencia)

2023	GPT-4	LLM	Amplió significativamente el razonamiento y las capacidades multimodales respecto a generaciones anteriores.	Solo en línea
2024	Gemini 1.5	LLM	Popularizó el contexto extremadamente largo y el procesamiento multimodal a gran escala.	Solo en línea
2024	Gemma 2	LLM	Familia abierta y eficiente orientada a investigación, personalización y despliegue con menores recursos.	Descargable (con licencia)
2024	Claude 3 / 3.5	LLM	Avanzó en razonamiento, programación, visión y procesamiento de contexto largo.	Solo en línea
2024	Llama 3 / 3.1	LLM	Acercó significativamente los modelos con pesos disponibles al rendimiento de modelos propietarios líderes.	Descargable (con licencia)
2024	Mistral / Mixtral	LLM	Consolidó arquitecturas MoE eficientes y modelos abiertos con buena relación entre rendimiento y costo computacional.	Descargable (uso libre)
2024	OpenAI o1	LLM	Consolidó el paradigma de modelos que emplean mayor cómputo durante la inferencia para resolver tareas complejas.	Solo en línea
2025	DeepSeek-R1	LLM	Popularizó modelos de razonamiento abiertos entrenados intensivamente con aprendizaje por refuerzo.	Descargable (uso libre)
2025	Gemini 2.5	LLM	Integró razonamiento avanzado, multimodalidad y contexto largo dentro de una misma generación.	Solo en línea
2025	Llama 4	LLM	Introdujo modelos nativamente multimodales basados en MoE, como Scout y Maverick.	Descargable (con licencia)
2025	Qwen3	LLM	Combina modos de razonamiento y respuesta rápida, con variantes densas y MoE.	Descargable (uso libre)
2025	Qwen3-Coder	LLM	Especializó Qwen3 en programación agéntica, uso de herramientas y generación de código.	Descargable (uso libre)
2025	GPT-5	LLM	Unificó en un solo sistema la respuesta rápida y el razonamiento profundo, eligiendo automáticamente cuánto pensar.	Solo en línea
2025	Mistral Large 3	LLM	Modelo multimodal MoE diseñado para combinar alta capacidad con eficiencia computacional.	Descargable (uso libre)
2025	Gemini 3 / 3.1 Pro	LLM	Elevó el razonamiento multimodal y la programación autónoma; 3.1 Pro duplicó el resultado de Gemini 3 Pro en ARC-AGI-2.	Solo en línea
2025–2026	Claude 4.x / 5	LLM	Evolucionó hacia razonamiento, programación, planificación agéntica y uso avanzado de herramientas.	Solo en línea
2026	Claude Sonnet 4.6	LLM	Orientado a programación, agentes, uso de computadora y trabajo del conocimiento con contexto muy extenso.	Solo en línea

2026	Mistral Small 4	LLM	Modelo MoE multimodal que combina instrucciones, razonamiento y programación en una arquitectura relativamente ligera.	Descargable (uso libre)
2026	DeepSeek-V4	LLM	MoE con atención comprimida que hace práctico el contexto de 1M para agentes, usando una fracción del cómputo y la memoria de V3.2.	Descargable (uso libre)
2026	Muse Spark	LLM	Primer modelo de Meta Superintelligence Labs; sucede a Llama con razonamiento multimodal propietario y un compañero abierto (Muse Glimmer).	Solo en línea (Muse Glimmer descargable)
2026	Claude Fable 5 / Mythos 5	LLM	Abrió el nivel Mythos de Anthropic; mismo modelo en dos versiones, Fable con salvaguardas adicionales y Mythos de acceso restringido.	Solo en línea (Mythos de acceso restringido)
2026	GPT-5.6 (Sol, Terra, Luna)	LLM	Priorizó la eficiencia por token con tres niveles de precio y capacidad que comparten el mismo contexto.	Solo en línea
2026	Qwen3.8-Max	LLM	MoE multimodal de 2,4 billones de parámetros (95.000 millones activos), el mayor modelo de Qwen hasta la fecha.	Descargable (anunciado)
2026	Gemini 3.8 Flash	LLM	Versión rápida centrada en ingeniería de software y tareas agénticas, con lanzamientos Flash muy frecuentes.	Solo en línea
2026	GPT-6 (Astra, Sol, Luna)	LLM	Nueva generación de razonamiento; Astra fue el primer modelo de OpenAI clasificado como riesgo crítico en ciberseguridad.	Solo en línea
2026	Claude Opus 5.5 / Sonnet 5.5	LLM	Orientados a programación agéntica y trabajo prolongado, con razonamiento siempre activo y profundidad ajustable.	Solo en línea
2026	Grok 4.7	LLM	Modelo orientado a razonamiento, programación y trabajo del conocimiento de larga duración.	Solo en línea

Acceso

- **Descargable (uso libre):** se puede descargar, modificar y usar incluso comercialmente (licencias permisivas como Apache 2.0 o MIT).
- **Descargable (con licencia):** se puede descargar y usar en el propio equipo, pero la licencia del creador impone condiciones.
- **Solo en línea:** solo se usa a través de la aplicación o la API de la empresa.
- **No publicado:** nunca estuvo disponible para el público.

Categoría

- LLM generativo moderno: modelo de lenguaje entrenado a gran escala que, en esta guía, se caracteriza por generación, seguimiento de instrucciones y capacidades generales o amplias.
- Modelo de representación / embeddings: comprende texto o lo convierte en vectores para clasificar, buscar o comparar; su salida principal no es una respuesta generativa libre.
- Modelo generativo precursor: genera texto y constituye un antecedente directo de los LLM generativos modernos; se incluye como categoría histórica y operativa de esta guía.

Modelo, aplicación, repositorio, API y plataforma

En el ecosistema actual de inteligencia artificial conviene distinguir varios elementos que suelen confundirse. Un **modelo** es la red entrenada que realiza una tarea, como GPT, Llama, Qwen, Mistral, BGE o SPECTER2. Una **aplicación** es la interfaz con la que interactúa directamente el usuario, como ChatGPT, Gemini o Claude.ai. Un **repositorio de modelos**, como Hugging Face Hub, permite publicar, descubrir, documentar y descargar modelos y sus archivos asociados. Una **API** permite que un programa se comunique con un modelo alojado en infraestructura externa. Un **proveedor** es la organización o infraestructura que ejecuta y sirve ese modelo. Una **plataforma agregadora**, como OpenRouter, facilita el acceso a múltiples modelos y proveedores mediante una interfaz común. Por otro lado, herramientas como Ollama, llama.cpp o vLLM permiten ejecutar modelos en infraestructura propia.

ELEMENTO	QUÉ REPRESENTA	EJEMPLOS
MODELO	Red neuronal entrenada para realizar determinadas tareas	GPT, Llama, Qwen, Mistral, BGE, SPECTER2
APLICACIÓN	Interfaz utilizada directamente por el usuario	ChatGPT, Gemini, Claude.ai
REPOSITORIO DE MODELOS	Plataforma para publicar, documentar y distribuir modelos	Hugging Face Hub
API	Mecanismo de comunicación entre un programa y un modelo remoto	OpenAI API, Anthropic API, Gemini API
PROVEEDOR	Empresa o infraestructura que ejecuta y sirve un modelo	OpenAI, Anthropic, Google, Mistral AI, Together AI
PLATAFORMA AGREGADORA	Acceso unificado a modelos y proveedores diferentes	OpenRouter
RUNTIME / MOTOR DE INFERENCIA	Software que carga y ejecuta modelos en infraestructura propia	Ollama, llama.cpp, vLLM
INTERFAZ LOCAL	Aplicación gráfica para gestionar y ejecutar modelos localmente	LM Studio

Ejecución mediante API frente a ejecución local

Un modelo puede ejecutarse principalmente de dos formas: mediante **infraestructura externa**, a la que se accede normalmente por una API, o mediante **infraestructura propia**, utilizando herramientas de inferencia local. La elección no cambia necesariamente la familia del modelo, sino el lugar donde se ejecuta, los recursos utilizados y el nivel de control disponible.

CRITERIO	MEDIANTE API	EJECUCIÓN LOCAL
INFRAESTRUCTURA	Servidores externos	Equipo o servidor propio
COSTO	Normalmente pago por uso o tokens	Hardware, energía y mantenimiento
PRIVACIDAD	Depende de las políticas del proveedor	Mayor control sobre los datos
FACILIDAD INICIAL	Alta	Requiere instalación y configuración
ESCALABILIDAD	Generalmente alta	Limitada por el hardware disponible
CONTROL TÉCNICO	Parcial	Alto
ACTUALIZACIÓN	Gestionada por el proveedor	Gestionada por el usuario
REPRODUCIBILIDAD	Puede cambiar entre versiones o proveedores	Mayor control de versión, cuantización y configuración

Hugging Face

Hugging Face (<https://huggingface.co/models>) es un ecosistema de inteligencia artificial cuyo componente más conocido, Hugging Face Hub, permite publicar, descubrir y reutilizar modelos, datasets y aplicaciones. Puede entenderse de forma simplificada como un repositorio colaborativo para modelos de IA, parecido al papel que GitHub cumple para el código.

Hugging Face no es el creador de todos los modelos que aloja. BERT, SciBERT, SPECTER, Llama o Qwen fueron desarrollados por distintas organizaciones y pueden publicarse en el Hub junto con sus pesos, configuración, tokenizer, licencia y documentación. La biblioteca Transformers permite descargar y ejecutar muchos de estos modelos desde Python, de forma local o mediante servicios de inferencia.

Su principal ventaja es reutilizar modelos preentrenados sin entrenarlos desde cero. Como precaución, antes de utilizar un modelo conviene revisar el autor, la licencia, la versión, la Model Card, sus limitaciones y los recursos de hardware necesarios. Además, un modelo base no siempre está listo para una tarea concreta: por ejemplo, SciBERT puede requerir una cabeza de clasificación y ajuste con datos etiquetados para convertirse en un clasificador específico.

Ejemplo de código en Python

```
from transformers import AutoModel
nombre_modelo = "google-bert/bert-base-uncased"
modelo = AutoModel.from_pretrained(nombre_modelo)

print("Modelo cargado correctamente")
print("Nombre:", nombre_modelo)
#print("Nombre:", modelo)
print("Tipo:", modelo.__class__.__name__)
print("Dimensión oculta:", modelo.config.hidden_size)
print("Número de capas:", modelo.config.num_hidden_layers)
```

Resultado

```
Modelo cargado correctamente
Nombre: google-bert/bert-base-uncased
Tipo: BertModel
Dimensión oculta: 768
Número de capas: 12
```

Nota:

La variable modelo está comentada en el código, ya que si se imprime, muestra en detalle la estructura interna del modelo BERT, entre ellas, arquitectura del modelo, número de vocabulario de tokens, los embeddings, el encoder, el número de capa Transforme y los componentes de atención.

Ejecución local de modelos: Ollama y otros runtimes

Los modelos también pueden ejecutarse en infraestructura propia sin depender de una API externa. Para ello se utilizan runtimes o motores de inferencia, que se encargan de cargar el modelo, gestionar sus archivos y ejecutar las solicitudes utilizando los recursos del equipo o servidor.

Ollama (<https://ollama.com/download/windows>) es una de las herramientas más accesibles para este propósito. Permite descargar, administrar y ejecutar modelos localmente mediante comandos sencillos y una API local. Puede utilizar modelos como Llama, Qwen, Mistral, Gemma o DeepSeek, siempre que exista una versión compatible. Una vez descargado el modelo, la inferencia puede realizarse sin enviar necesariamente los datos a un proveedor externo.

Esta plataforma, ejecuta modelos y permite trabajar tanto con modelos descargados y ejecutados localmente como, en algunas configuraciones, con modelos disponibles mediante servicios cloud. En el selector de modelos, los modelos locales se ejecutan utilizando los recursos del equipo, mientras que los identificados como cloud utilizan infraestructura remota.

La imagen 1, muestra la interfaz de Ollama Desktop y la presencia de un modelo en el selector no implica necesariamente que esté instalado localmente; conviene verificar si aparece como descargable, instalado o identificado como modelo cloud.

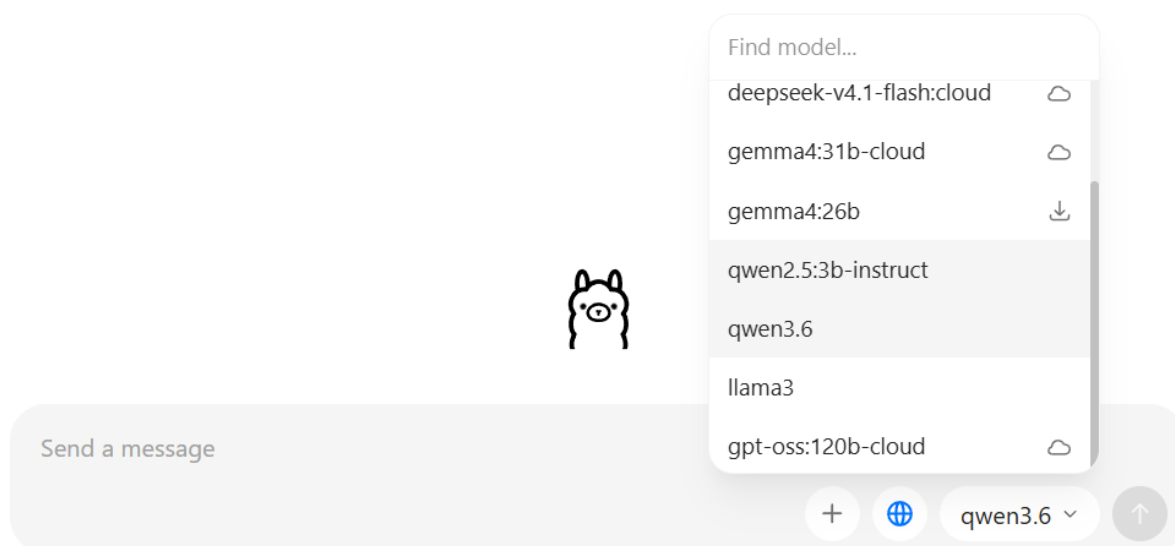


Imagen 1. Ventana de Ollama Desktop y modelos integrados

OpenRouter

OpenRouter es una plataforma de API unificada que permite a los desarrolladores acceder a cientos de modelos de inteligencia artificial como muestra la imagen 2, que son de múltiples proveedores, estos se acceden a través de un único punto final y con una sola clave. Funciona como un proxy inteligente que traduce las solicitudes de tu aplicación, esta da acceso a más de 400 modelos de distintos creadores mediante una sola API, con el mismo código y una sola cuenta de pago ([Models](#)).

Cómo funciona

- **Suscribirse a la Plataforma**
- **Adquirir créditos para el uso de los modelos**
- **Una API para todos los modelos:** se cambia de modelo cambiando solo su identificador, con el formato creador/modelo. Ejemplos: `mistralai/mistral-small-3.2-24b-instruct` o `deepseek/deepseek-v4-flash`.
- **Modelo frente a proveedor:** un mismo modelo puede estar servido por varios proveedores (empresas que lo ejecutan en sus servidores). OpenRouter elige el proveedor en cada solicitud y, si uno falla, pasa al siguiente.
- **Precios:** OpenRouter cobra el mismo precio por token que el proveedor, sin recargo sobre la inferencia ([FAQ](#)). Se paga por adelantado con créditos, la compra con tarjeta tiene una comisión del 5,5% (mínimo 0,80 USD).
- **Modelos gratuitos:** los que llevan (free) o el sufijo :free no cuestan nada, pero tienen un límite de 50 solicitudes diarias, o de 1.000 si se compraron al menos 10 USD en créditos.
- **Privacidad por defecto:** OpenRouter guarda solo metadatos de cada solicitud, no el texto de los prompts ni de las respuestas, salvo que el usuario active el registro.
- **Dashboard de actividad:** puedes ver cuanto token consumes, cuantas peticiones y cuál es el costo de consumo por cada modelo que utilizas.

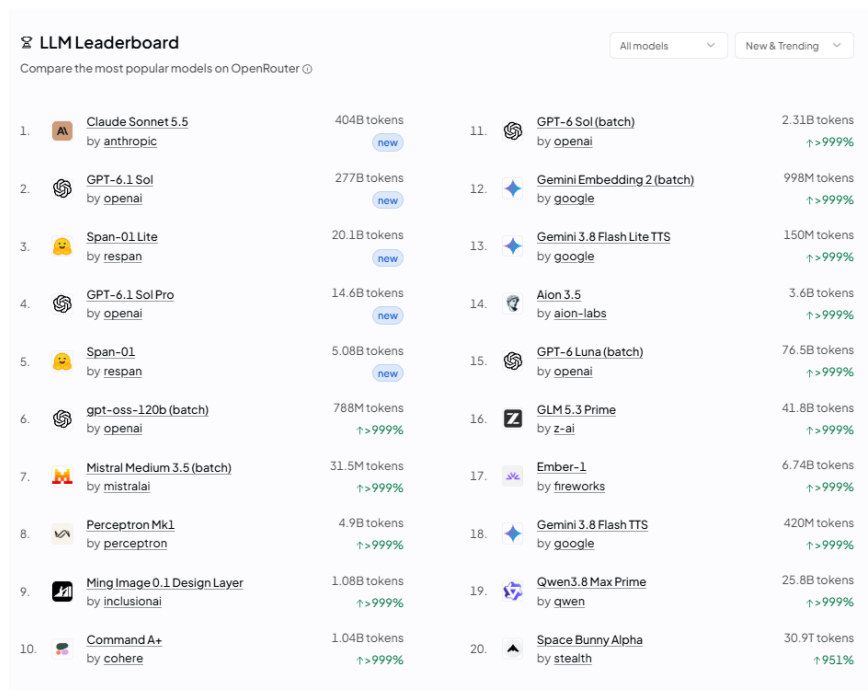
All	T Text 464	Image 57	Video 30	Speech 23	Decisions 9	Embeddings 37	Rerank 9	Transcription 0	Audio 4
ByteDance Seed: Seedream 5.0 Flash									
Seedream 5.0 Flash is an image generation and editing model from ByteDance Seed. It is the fast, cost-efficient tier of the Seedream 5.0 family, suited for high-volume production and interactive editing workflows that need precise edits at low latency.									
by bytedance-seed Oct 1, 2026 from \$0.018/image									
Black Forest Labs: FLUX.3 Image									
FLUX.3 Image is Black Forest Labs' flagship image generation and editing model. It handles text-to-image and multi-reference editing with up to 10 input images, and renders at fixed resolution tiers from 768 up to 4K with a selectable aspect ratio. Pricing is a flat per-image rate that scales with the chosen resolution tier.									
by black-forest-labs Oct 1, 2026 47K context from \$0.0205/image									
LiquidAI: D1									
D1 is Liquid AI's structured decision model, served as a System One endpoint. Send a state along with typed questions, and it returns a choice, a score, or a yes/no answer, each with a probability taken directly from the model rather than written out as text. It uses the same /v1/systemone schema as other OpenRouter Decisions models, so it suits routing, classification, and policy checks ...									
by liquid Oct 1, 2026 66K context \$0.04/M input tokens \$0/M output tokens									
Apodex: Apodex 1.1 Mini (free)									
Apodex 1.1 Mini is a reasoning-first model from Apodex, built for complex, long-horizon research and forecasting tasks. It works directly with files, data, code, and tools to produce verifiable results, and is designed for agentic research workflows where answers need to be grounded in evidence.									
by apodex Oct 1, 2026 262K context \$0/M input tokens \$0/M output tokens									

Imagen 2. Listado de modelos y característica sen OpenRouter

LLM Leaderboard de OpenRouter

El **LLM LeaderBoard** es una lista que clasifica los modelos de inteligencia artificial según el volumen real de tokens procesados por los desarrolladores a través de su plataforma. Un LLM leaderboard no mide **qué modelo es mejor** de manera universal; mide un aspecto concreto del desempeño según una metodología determinada.

A diferencia de otros marcadores basados en exámenes teóricos o pruebas de rendimiento académico, este ranking mostrado en la imagen 3, mide la cantidad de datos (tokens) que las aplicaciones y usuarios mueven en producción, creando un rankind de los más utilizados.



Rank	Model	Provider	Tokens	Change
1.	Claude Sonnet 5.5	by anthropic	404B tokens	new
2.	GPT-6.1 Sol	by openai	277B tokens	new
3.	Span-01 Lite	by respan	20.1B tokens	new
4.	GPT-6.1 Sol Pro	by openai	14.6B tokens	new
5.	Span-01	by respan	5.08B tokens	new
6.	gpt-oss-120b (batch)	by openai	788M tokens	↑ >999%
7.	Mistral Medium 3.5 (batch)	by mistralai	31.5M tokens	↑ >999%
8.	Perceptron Mk1	by perceptron	4.9B tokens	↑ >999%
9.	Ming Image 0.1 Design Layer	by inclusionai	1.08B tokens	↑ >999%
10.	Command A+	by cohere	1.04B tokens	↑ >999%
11.	GPT-6 Sol (batch)	by openai	2.31B tokens	↑ >999%
12.	Gemini Embedding 2 (batch)	by google	998M tokens	↑ >999%
13.	Gemini 3.8 Flash Lite TTS	by google	150M tokens	↑ >999%
14.	Alon 3.5	by alon-labs	3.6B tokens	↑ >999%
15.	GPT-6 Luna (batch)	by openai	76.5B tokens	↑ >999%
16.	GLM 5.3 Prime	by z-ai	41.8B tokens	↑ >999%
17.	Ember-1	by fireworks	6.74B tokens	↑ >999%
18.	Gemini 3.8 Flash TTS	by google	420M tokens	↑ >999%
19.	Qwen3.8 Max Prime	by qwen	25.8B tokens	↑ >999%
20.	Space Bunny Alpha	by stealth	30.9T tokens	↑ 951%

Imagen 3. LLM Leaderboard de OpenRouter a 01 de octubre de 2026

Otros LeaderBoard

- LLM LeaderBoard OpenRouter - <https://openrouter.ai/rankings?view=trending#top-models>
- Artificial Analysis – LLM Leaderboard
https://artificialanalysis.ai/leaderboards/models/?utm_source=chatgpt.com
- Hugging Face Open LLM Leaderboard - <https://huggingface.co/spaces/open-llm-leaderboard/open-llm-leaderboard?theme=light#/>
- Berkeley Function Calling Leaderboard (BFCL) –
https://gorilla.cs.berkeley.edu/leaderboard?utm_source=chatgpt.com
- SEAL Leaderboards – Scale AI - https://labs.scale.com/leaderboard?utm_source=chatgpt.com
- WebDev Arena- <https://arena.ai/leaderboard>
- Aider LLM Leaderboard - https://aider.chat/docs/leaderboards/?utm_source=chatgpt.com

Los modelos más usados no son los mejor puntuados

Puesto de cada modelo (su mejor variante) en cinco clasificaciones activas, a 1-4 de octubre de 2026

		OpenRouter Uso (tokens, 30 días)	Artificial Analysis Índice de inteligencia	Arena Texto	Arena WebDev	SEAL · Scale AI Humanity's Last Exam	PUESTO EN LA CLASIFICACIÓN
ANTHROPIC	Claude Opus 5.5	—	1	4	1	—	1-3
	Claude Sonnet 5.5	—	2	—	3	—	4-8
	Claude Fable 5.1	—	5	6	5	2	9-15
OPENAI	GPT-6 Astra	—	7	—	2	1	— Fuera del top 15 o no evaluado
	GPT-6.1 Sol	—	11	—	4	—	
	GPT-5.6 Luna	5	—	—	—	—	
GOOGLE	Gemini 4 Argon	—	8	1	9	—	
	Gemini 3.8 Flash	—	—	8	—	2	
META	Muse Spark 1.3	—	—	9	14	4*	
ALIBABA	Qwen3.8 Max	—	—	—	10	—	
XAI	Grok 4.7	—	—	—	15	—	
MOONSHOT	Kimi K3	—	—	—	13	—	
DEEPSEEK	DeepSeek V4.1 Flash	1	—	—	—	—	
Z.AI	GLM 5.3	2	—	—	—	—	
XIAOMI	MiMo-V2.6	10	—	—	—	—	

LECTURA

El uso real en OpenRouter lo encabezan modelos baratos y rápidos (DeepSeek, GLM, MiMo). Las clasificaciones de calidad las lideran Claude, GPT-6 y Gemini. Ningún modelo destaca en las cinco: cada ranking mide algo distinto.

FUERA DE LA MATRIZ

Berkeley Function Calling (BFCL v4)
Mide llamada a funciones. Lo lideran modelos pequeños y abiertos (BTL-3, 88,5 %); ninguno de los de la matriz está en su top 15.

Aider Polyglot
Mide programación. Sin actualizar desde el 25 ago 2025 (líder: GPT-5 high, 88 %).

Hugging Face Open LLM Leaderboard
Retirado el 13 mar 2025; ya no clasifica modelos actuales.

Cada celda indica el puesto de la mejor variante del modelo en el top 15 de cada clasificación (top 10 en OpenRouter, top 20 en SEAL). OpenRouter: tokens procesados en los últimos 30 días. Artificial Analysis: Intelligence Index. Arena: votos humanos en texto y desarrollo web. SEAL: Humanity's Last Exam. * Muse Spark (versión base). BFCL según benchlm.ai (2 oct 2026). Fuentes: openrouter.ai/rankings · artificialanalysis.ai · arena.ai · labs.scale.com · gorilla.cs.berkeley.edu · aider.chat · huggingface.co

Características del catálogo de modelos en OpenRouter

La página de modelos permite filtrar el catálogo por estas características, donde cada modelo muestra el creador, la fecha de publicación, la ventana de contexto, el precio y los tokens procesados en la plataforma (un indicador de popularidad).

Característica	Qué significa	Ejemplo
Tipo de salida (se muestra en el menú del listado de modelos)	Qué produce el modelo: texto, imagen, video, voz, decisiones, embeddings, reordenación, transcripción o audio	La pestaña Embeddings reúne los modelos para búsqueda semántica
Modalidad de entrada (<i>Input modalities</i>)	Qué acepta el modelo como entrada: texto, imagen, archivo, audio o video	Para leer PDF escaneados se filtra por Image o File
Ventana de contexto (<i>Context length</i>)	Máximo de tokens por solicitud, de 4K a 1M en el filtro	Una taxonomía larga con 50 títulos por lote necesita al menos 32K
Precio de entrada (<i>Prompt pricing</i>)	Costo por millón de tokens enviados al modelo	0,04 USD por millón de tokens de entrada en LiquidAI D1
Precio de salida (<i>Output pricing</i>)	Costo por millón de tokens generados; suele ser varias veces el de entrada	Una salida JSON compacta reduce este costo
Serie (<i>Series</i>)	Familia del modelo: GPT, Claude, Gemini, Gemma, Grok, Qwen, DeepSeek, Mistral, Llama...	Para comparar versiones de Qwen se filtra por la serie Qwen
Categorías (<i>Categories</i>)	Uso para el que destaca el modelo: programación, rol, traducción, ciencia, etc.	La categoría de programación agrupa modelos como Qwen3-Coder
Parámetros admitidos (<i>Supported parameters</i>)	Opciones que acepta el modelo en la solicitud: temperature, top_k, seed, tools, response_format, structured_outputs, reasoning.	Para clasificar con top_k = 1 y salida JSON se exigen ambos parámetros
Llamada a herramientas (<i>Tool calling</i>)	El modelo puede invocar funciones externas, como una búsqueda o una base de datos	Consultar OpenAlex antes de clasificar un artículo
Destilable (<i>Distillable</i>)	El creador permite usar las respuestas del modelo para entrenar otros modelos	Usar un LLM grande como <i>teacher</i> para etiquetar 45.000 documentos y entrenar un modelo local
Retención cero (<i>Zero data retention, ZDR</i>)	El proveedor no guarda los datos enviados ni un instante; tampoco puede entrenar con ellos	Necesario si los documentos son confidenciales
Enrutamiento en región (<i>In-region routing</i>)	Los datos se procesan solo dentro de una región, como la UE o EE. UU.	Una institución europea que debe cumplir el RGPD
Antigüedad (<i>Model age</i>)	Fecha de publicación del modelo en la plataforma	Descartar modelos de más de un año
Modelos inactivos (<i>Inactive models</i>)	Muestra u oculta modelos retirados o sin proveedor activo	Útil para reproducir un experimento antiguo
Artificial Analysis	Puntuaciones de capacidad y velocidad de esta evaluadora independiente	Ordenar candidatos por calidad antes de probarlos
Design Arena	Clasificación por votos en tareas de diseño e interfaces	Elegir un modelo para generar maquetas web
Proveedores (<i>Providers</i>)	Empresas que sirven el modelo; difieren en precio, velocidad y políticas de datos	Mistral Small servido por Mistral o por otro proveedor
Autores (<i>Model authors</i>)	Organización que creó el modelo	Filtrar por deepseek o qwen
Variantes (<i>All variants</i>)	Versiones del mismo modelo con sufijo: :free, :thinking, :nitro, :floor, :online, :exacto, :extended	:nitro prioriza velocidad :floor, el precio más bajo

Parámetros de generación de un LLM

Durante la inferencia, un modelo generativo calcula una distribución de probabilidades para los posibles tokens siguientes. Los parámetros de generación modifican la forma en que se seleccionan esos tokens y, por tanto, influyen en la variabilidad, creatividad, repetición y reproducibilidad de la respuesta. Estos parámetros pueden configurarse tanto en modelos ejecutados localmente como mediante APIs, aunque su disponibilidad y comportamiento exacto dependen del modelo, runtime y proveedor. En OpenRouter, el filtro **Supported parameters** muestra qué modelos admiten cada parámetro.

Como ejemplo, ante el texto “**El artículo trata sobre...**”, se generan los siguientes datos donde se usa una distribución de referencia, el modelo asigna estas probabilidades al **siguiente token**:

- Biología = 0.40
- Medicina = 0.25
- Química = 0.15
- Física = 0.10
- Historia = 0.06
- cocina = 0.03
- otras = 0.01

Qué representa

Un LLM no escribe una frase de golpe, produce una palabra (un token) cada vez. Cuando ha leído “**El artículo trata sobre...**”, calcula para cada palabra posible qué tan probable es que sea la siguiente.

- Hay un 40 % de probabilidad de que siga **biología**.
- Un 25 % de que siga **medicina**.
- Así sucesivamente, hasta **otras**, que agrupa el 1 % restante.

Qué hacen los parámetros con esa tabla

Cambian cómo se elige una palabra a partir de esas probabilidades, en el parámetro de **temperature** cambia la forma de la distribución.

- Con **0**, siempre se elige la más probable: **biología**.
- Con 0,5, la distribución se concentra y **biología** sube a ~62 %.
- Con 1, se respetan los valores tal cual: **biología** saldría ~4 de cada 10 veces y **medicina** ~2,5.
- Con 1,5, la distribución se aplana: **biología** baja a ~31 % e **historia** sube del 6 % al ~9 %.

Nota: Las siete cifras del ejemplo, suman 1 (el 100 %), porque juntas cubren todas las opciones. En un modelo real la lista no tiene siete opciones, sino todo su vocabulario, que suele ser de 100.000 a 200.000 tokens. Casi todos tienen una probabilidad pequeña.

Tabla de Significado de los parámetros

Parámetro	Qué controla	Al disminuirlo / valor neutro	Al aumentarlo	Ejemplo con la distribución de referencia
temperature	Grado de aleatoriedad al elegir el siguiente token	0: respuestas casi deterministas y conservadoras	Más variedad y creatividad, pero mayor riesgo de incoherencia	0 → siempre “biología”; 1 → “biología” ~40 % y “medicina” ~25 %; 1,5 → la distribución se aplanar y “historia” o “cocina” aparecen más a menudo
top_k	Número máximo de tokens candidatos considerados	1: solo el token más probable (greedy)	Más candidatos y más diversidad	1 → solo “biología”; 3 → elige entre biología, medicina y química
top_p	Probabilidad acumulada de los candidatos (nucleus sampling)	Valores bajos: solo los tokens más probables	1: no descarta ningún token	0,5 → biología + medicina (0,65); 0,9 → los cuatro primeros; 1 → ninguno descartado
min_p	Probabilidad mínima relativa al token más probable	0: sin filtro	Elimina tokens poco probables	0,1 → umbral 0,04: elimina cocina y otras; 0,2 → umbral 0,08: quedan los cuatro primeros
seed	Semilla del generador aleatorio	Sin semilla: cada ejecución puede variar	Un valor fijo favorece la reproducibilidad	Con temperature 0,7 y sin seed, dos ejecuciones pueden diferir; con seed 12345 tienden a coincidir
frequency_penalty	Penaliza tokens según cuántas veces ya aparecieron	0: sin penalización	Reduce repeticiones de palabras	0,5 evita repetir “aprendizaje” en una lista de 10 palabras clave; perjudica la clasificación por lotes
presence_penalty	Penaliza tokens que ya aparecieron al menos una vez	0: sin penalización	Favorece temas y palabras nuevas	0,5 empuja a proponer temas distintos en una lluvia de ideas
repetition_penalty	Penalización multiplicativa de la repetición	1: sin efecto	Valores > 1 reducen bucles y repeticiones	1.1 puede ayudar cuando un modelo repite frases.

Perfiles orientativos por tipo de tarea (puntos de partida)

TAREA	TEMPERATURE	TOP_P	PENALIZACIONES
CLASIFICACIÓN Y EXTRACCIÓN	0 (top_k 1)	1	0 (repetition_penalty 1)
RESUMEN Y TRADUCCIÓN	0,2–0,3	0,9	0
REDACCIÓN GENERAL	0,7	0,9	0–0,3
LLUVIA DE IDEAS	0,9–1,1	0,95	presence 0,3–0,6

Ejemplo: configuración reproducible en local y en OpenRouter

Los mismos valores se usaron con Mistral tanto en ejecución local como a través de OpenRouter, para obtener respuestas estables en tareas de clasificación:

temperature 0 · top_k 1 · top_p 1 · min_p 0 · seed 12345 · frequency_penalty 0

Local con Ollama. Los parámetros van dentro del objeto options:

```
import requests

datos = {
    "model": "qwen3.6",
    "prompt": "Explica qué es un Transformer",
    "stream": False,
    "options": {
        "temperature": 0,
        "top_k": 1,
        "top_p": 1,
        "min_p": 0,
        "seed": 12345,
        "frequency_penalty": 0
    }
}

r = requests.post("http://localhost:11434/api/generate", json=datos)

if r.status_code == 200:
    print(r.json()["response"])
else:
    print("Error:", r.status_code)
    print(r.text)
```

Resultado

¿Qué es un Transformer?

Un **Transformer** es un modelo de arquitectura de redes neuronales profundas introducido en el artículo **"Attention Is All You Need"** (2017) por Vaswani et al. Diseñado originalmente para traducción automática, se ha convertido en la base de la mayoría de los modelos de inteligencia artificial moderna, incluidos los LLMs como GPT, BERT y Claude.

Su innovación principal es reemplazar las capas recurrentes (RNN/LSTM) y convolucionales por un mecanismo llamado **Autoatención (Self-Attention)**, que permite procesar todos los elementos de una secuencia en paralelo y capturar dependencias a larga distancia de manera eficiente.

📌 Componentes Clave

1. **Mecanismo de Autoatención (Self-Attention)**
 - * Calcula un peso para cada token en relación con todos los demás tokens de la entrada.
 - * Permite que el modelo "preste atención" a palabras relevantes independientemente de su distancia, resolviendo el problema de dependencia a largo plazo de las RNNs.
 - * Se implementa típicamente como **Multi-Head Attention**, donde múltiples cabezas aprenden diferentes tipos de relaciones (sintácticas, semánticas, etc.) en paralelo.

API de OpenRouter. Los parámetros van en el nivel superior de la solicitud:

```
import requests

TOKEN = "xxxxxxxxxxxxxxxxxxxx" # clave de OpenRouter

datos = {
    "model": "mistralai/mistral-small-3.2-24b-instruct",
    "messages": [
        {"role": "user",
         "content": "Clasifica: Deep learning for breast cancer detection"}
    ],
    "temperature": 0,
    "top_k": 1,
    "top_p": 1,
    "min_p": 0,
    "seed": 12345,
    "frequency_penalty": 0,
    "presence_penalty": 0,
    "repetition_penalty": 1
}

r = requests.post("https://openrouter.ai/api/v1/chat/completions",
                  headers={"Authorization": f"Bearer {TOKEN}"},
                  json=datos)

print(r.json()["choices"][0]["message"]["content"])
```

Resultado

El **Deep Learning (DL)** para la detección de cáncer de mama se clasifica en varias categorías según su enfoque, tipo de datos utilizados y aplicación. Aquí tienes una clasificación detallada:

```
### **1. Por tipo de datos de entrada**
- Imágenes de mamografía (2D/3D):
  - Redes convolucionales (CNNs) para análisis de imágenes digitales (ej. HeMIS, DeepLesion).
  - Detección de microcalcificaciones, masas y distorsiones arquitectónicas.
- Imágenes de ultrasonido:
  - CNNs para clasificación de lesiones (benignas/malignas).
  - Redes híbridas (CNN + LSTM) para secuencias temporales.
- Resonancia magnética (MRI):
  - Redes 3D (ej. 3D U-Net) para segmentación de tumores.
  - Modelos de atención para focalizar en áreas sospechosas.
- Tomosíntesis mamaria (DBT):
  - Redes 3D para reconstrucción y detección en capas.
- Imágenes histopatológicas (biopsias):
  - Redes como ResNet o EfficientNet para clasificación de tejidos.
```

Notas. Con `top_k = 1` los demás filtros y la semilla dejan de influir en la salida, pero fijarlos es una buena práctica porque no todos los proveedores respetan `top_k`. Las penalizaciones se mantienen en 0 (`repetition_penalty` en 1) para no castigar etiquetas que se repiten legítimamente. En Ollama, `repetition_penalty` se llama `repeat_penalty`. La clave de la API se coloca en el código a manera de práctica, pero debe guardarse en una variable de entorno.

Qué evaluar para seleccionar un modelo

El mejor modelo no es el más potente, sino el que resuelve la tarea con calidad suficiente, velocidad, al menor costo y con las garantías de privacidad y reproducibilidad que exige el proyecto.

Criterios de selección

Criterio	Pregunta que responde	Dónde se ve en OpenRouter
Tipo de tarea	¿Necesito generar texto, buscar documentos parecidos o solo clasificar?	Pestañas Text, Embeddings, Rerank, Decisions
Calidad	¿Distingue bien categorías cercanas? Se mide con una muestra validada a mano	Artificial Analysis y pruebas propias
Costo	¿Cuánto cuesta procesar todo el corpus, sumando entrada y salida?	Prompt pricing y Output pricing
Ventana de contexto	¿Cabén las instrucciones, la taxonomía, el lote y la respuesta?	Context length
Modalidad	¿Los documentos tienen imágenes, tablas escaneadas o audio?	Input modalities
Formato de salida	¿Devuelve JSON válido de forma estable?	Supported parameters: response_format, structured_outputs
Reproducibilidad	¿Da el mismo resultado al repetir? ¿Admite temperature, top_k y seed?	Supported parameters; fijar proveedor con order
Velocidad	¿Cuánto tarda cada lote y todo el corpus?	Throughput y latencia de cada proveedor
Privacidad	¿Pueden los datos salir de la institución? ¿Se guardan?	Zero data retention, In-region routing, data_collection
Acceso	¿Necesito ejecutarlo en local o me basta una API?	Columna Acceso de los anexos
Uso de las salidas	¿Usaré las respuestas para entrenar otro modelo?	Distillable
Herramientas	¿Debe consultar fuentes externas durante la tarea?	Tool calling

Ejemplos de selección de Modelo

Los mismos criterios conducen a decisiones distintas según la necesidad. Los dos ejemplos siguientes aplican la tabla anterior a dos tareas habituales en un entorno de investigación.

Ejemplo 1. Clasificar 37.000 artículos científicos en una taxonomía temática.

Un equipo debe asignar a cada artículo, a partir de su título y resumen, una subárea de una taxonomía de unas 300 etiquetas. Los datos son públicos y el resultado debe poder repetirse.

Ejemplo 2. Encontrar trabajos similares en un repositorio de tesis no publicadas.

Una biblioteca universitaria quiere recomendar, para cada tesis, las cinco más parecidas de su repositorio. Los textos son inéditos y no pueden salir de la institución.

Tabla de comparación de los dos ejemplos por criterios

Criterio	Ejemplo 1. Clasificar artículos en una taxonomía	Ejemplo 2. Buscar tesis similares
Tipo de tarea	Asignar una etiqueta siguiendo instrucciones y una taxonomía: se necesita un LLM (pestaña Text).	Encontrar documentos parecidos: basta un modelo de embeddings.
Calidad	Validar a mano 200–500 artículos y medir el acierto por subárea.	Revisar si las recomendaciones de una muestra de tesis son pertinentes.
Costo	Unos 741 lotes de 50 artículos: ~16 millones de tokens de entrada y ~0,6 millones de salida. Con un precio ilustrativo de 0,10 USD por millón de entrada y 0,30 USD de salida, unos 2 USD.	Cero: el modelo se ejecuta en un equipo propio.
Ventana de contexto	Taxonomía (~8K tokens) + 50 títulos y resúmenes (~12K) + respuesta: ~22K. Sirve cualquier modelo de 32K o más.	512 tokens por documento bastan para título y resumen.
Modalidad	Solo texto	Solo texto.
Formato de salida	JSON compacto y estable	No aplica: el modelo devuelve vectores.
Reproducibilidad	temperature 0, top_k 1, seed fijo	Total: el mismo texto produce siempre el mismo vector.
Velocidad	Proveedor con buen throughput (velocidad a la que el modelo genera texto), lotes de 50 y modo resume.	Minutos para todo el repositorio en un portátil.
Privacidad	Datos públicos: la API es aceptable	Textos inéditos: no pueden salir de la institución.
Acceso	API (OpenRouter).	Descargable (Hugging Face), ejecución local.
Uso de las salidas	Si las etiquetas servirán después para entrenar SciBERT, elegir un modelo Distillable.	No aplica.
Herramientas	No, la taxonomía ya va en el prompt.	No.

Decisión en el ejemplo 1

Conviene un LLM instruct no razonador, de tamaño medio, con soporte de salida estructurada, como por ejemplo de las familias Mistral Small, Qwen3 o DeepSeek-V4 en OpenRouter. Si las etiquetas se usarán para entrenar otro modelo, hay que comprobar que sea destilable (permite usar sus respuestas para entrenar otro modelo). Un modelo razonador o multimodal encarecería la corrida sin mejorar la clasificación.

Decisión en el ejemplo 2

No hace falta un LLM. Un modelo de embeddings científico como SPECTER2, descargado de Hugging Face y ejecutado en local, resuelve la tarea gratis, sin que los textos salgan de la institución y con resultados reproducibles.

En ambos casos la regla es la misma, empezar por la tarea y sus restricciones de privacidad, costo y reproducibilidad, y solo después elegir el modelo concreto.

Referencias

Red Hat. (s. f.). ¿Qué son los modelos de lenguaje de gran tamaño (LLM)?

<https://www.redhat.com/es/topics/ai/what-are-large-language-models>

IBM. (s. f.). ¿Qué son los grandes modelos de lenguaje (LLM)? IBM Think. [https://www.ibm.com/es-](https://www.ibm.com/es-es/think/topics/large-language-models)

[https://www.ibm.com/es-](https://www.ibm.com/es-es/think/topics/large-language-models)

Hugging Face. (s. f.). Transformers documentation. <https://huggingface.co/docs/transformers/index> x

Hugging Face. (s. f.). Hugging Face Hub documentation. <https://huggingface.co/docs/hub/index>

GuruSup. (s. f.). Modelos de lenguaje LLM. <https://gurusup.com/es/blog/modelos-de-lenguaje-llm>

Databricks. (s. f.). What are large language models (LLMs)? [https://www.databricks.com/es/blog/what-are-](https://www.databricks.com/es/blog/what-are-large-language-models)
[large-language-models](https://www.databricks.com/es/blog/what-are-large-language-models)

BBVA. (s. f.). Los LLM, modelos de lenguaje: qué son y cómo funcionan. BBVA Innovación.

<https://www.bbva.com/es/innovacion/los-llm-modelos-de-lenguaje-que-son-y-como-funcionan/>

Amazon Web Services. (s. f.). ¿Qué es un modelo de lenguaje grande (LLM)? AWS.

<https://aws.amazon.com/es/what-is/large-language-model/>

Alibaba Cloud. (2026, 3 de agosto). *Alibaba unveils Qwen3.8-Max: Its largest and most capable flagship model to date.* <https://www.alibabacloud.com/en/press-room/alibaba-unveils-qwen3-8-max>

Amazon Web Services. (2026, 3 de agosto). *OpenAI GPT-5.6 Sol, Terra, and Luna now support 1 million token context windows on Amazon Bedrock.* [https://aws.amazon.com/about-aws/whats-new/2026/08/gpt-sol-terra-](https://aws.amazon.com/about-aws/whats-new/2026/08/gpt-sol-terra-luna-long-context-bedrock/)
[luna-long-context-bedrock/](https://aws.amazon.com/about-aws/whats-new/2026/08/gpt-sol-terra-luna-long-context-bedrock/)

Anthropic. (s. f.). *Introducing Claude Fable 5 and Claude Mythos 5.* Claude Platform Docs. Recuperado el 2 de octubre de 2026, de [https://platform.claude.com/docs/en/about-claude/models/introducing-claude-fable-5-](https://platform.claude.com/docs/en/about-claude/models/introducing-claude-fable-5-and-claude-mythos-5)
[and-claude-mythos-5](https://platform.claude.com/docs/en/about-claude/models/introducing-claude-fable-5-and-claude-mythos-5)

Anthropic. (2026, 17 de febrero). *Introducing Claude Sonnet 4.6.* [https://www.anthropic.com/news/claude-](https://www.anthropic.com/news/claude-sonnet-4-6)
[sonnet-4-6](https://www.anthropic.com/news/claude-sonnet-4-6)

BenchLM. (2026). *Mistral model releases: Confirmed timeline & sources.* [https://benchlm.ai/model-](https://benchlm.ai/model-updates/providers/mistral)
[updates/providers/mistral](https://benchlm.ai/model-updates/providers/mistral)

CellCog. (2026). *Claude Sonnet 5.5: Released Sept 28, price, benchmarks.* [https://cellcog.ai/blog/claude-](https://cellcog.ai/blog/claude-sonnet-5-5-release-date/)
[sonnet-5-5-release-date/](https://cellcog.ai/blog/claude-sonnet-5-5-release-date/)

Digital Applied. (2026). *AI model releases: September 2026 tracker and dated ledger.*

<https://www.digitalapplied.com/blog/ai-model-releases-september-2026-tracker>

DutchStartup.ai. (2026). *Four major AI labs launch new models in the first week of September 2026.*

[https://www.dutchstartup.ai/en/news/four-major-ai-labs-launch-new-models-in-the-first-week-of-september-](https://www.dutchstartup.ai/en/news/four-major-ai-labs-launch-new-models-in-the-first-week-of-september-2026)
[2026](https://www.dutchstartup.ai/en/news/four-major-ai-labs-launch-new-models-in-the-first-week-of-september-2026)

Evolink. (2026). *Claude Opus 5.5 release date & what changed.* [https://evolink.ai/blog/claude-opus-5-5-](https://evolink.ai/blog/claude-opus-5-5-release-date)
[release-date](https://evolink.ai/blog/claude-opus-5-5-release-date)

Gemini 3 (AI). (s. f.). En *Wikipedia*. Recuperado el 2 de octubre de 2026, de [https://en.wikipedia.org/wiki/Gemini_3_\(AI\)](https://en.wikipedia.org/wiki/Gemini_3_(AI))

Guía práctica para seleccionar, configurar y evaluar modelos LLM en tareas de clasificación documental [Documento interno no publicado]. (2026).

Hugging Face. (2026). *DeepSeek-V4: A million-token context that agents can actually use*. <https://huggingface.co/blog/deepseekv4>

Lindy. (2026). *GPT-6 explained: Astra, Sol, Luna, and what each one costs*. <https://www.lindy.ai/blog/gpt-6>

Malik, U. (2026). *GPT-5.6 API: Pricing, thinking modes, and the shared context trap*. <https://umesh-malik.com/blog/openai-gpt-5-6-sol-terra-luna-guide>

Muse Spark. (s. f.). En *Wikipedia*. Recuperado el 2 de octubre de 2026, de https://en.wikipedia.org/wiki/Muse_Spark

OpenRouter. (s. f.-a). *Distillation*. Recuperado el 2 de octubre de 2026, de <https://openrouter.ai/docs/guides/guides/distillation>

OpenRouter. (s. f.-b). *FAQ*. Recuperado el 2 de octubre de 2026, de <https://openrouter.ai/docs/faq>

OpenRouter. (s. f.-c). *Models: Unified access to 400+ AI models*. Recuperado el 2 de octubre de 2026, de <https://openrouter.ai/docs/guides/overview/models>

OpenRouter. (s. f.-d). *Provider routing: Smart multi-provider request management*. Recuperado el 2 de octubre de 2026, de <https://openrouter.ai/docs/guides/routing/provider-selection>

OpenRouter. (s. f.-e). *Sovereign AI: In-region AI routing with OpenRouter*. Recuperado el 2 de octubre de 2026, de <https://openrouter.ai/docs/guides/features/sovereign-ai>

OpenRouter. (s. f.-f). *Structured outputs: Type-safe JSON responses from AI models*. Recuperado el 2 de octubre de 2026, de <https://openrouter.ai/docs/guides/features/structured-outputs>

OpenRouter. (s. f.-g). *Zero data retention: How OpenRouter gives you control over your data*. Recuperado el 2 de octubre de 2026, de <https://openrouter.ai/docs/guides/features/zdr>