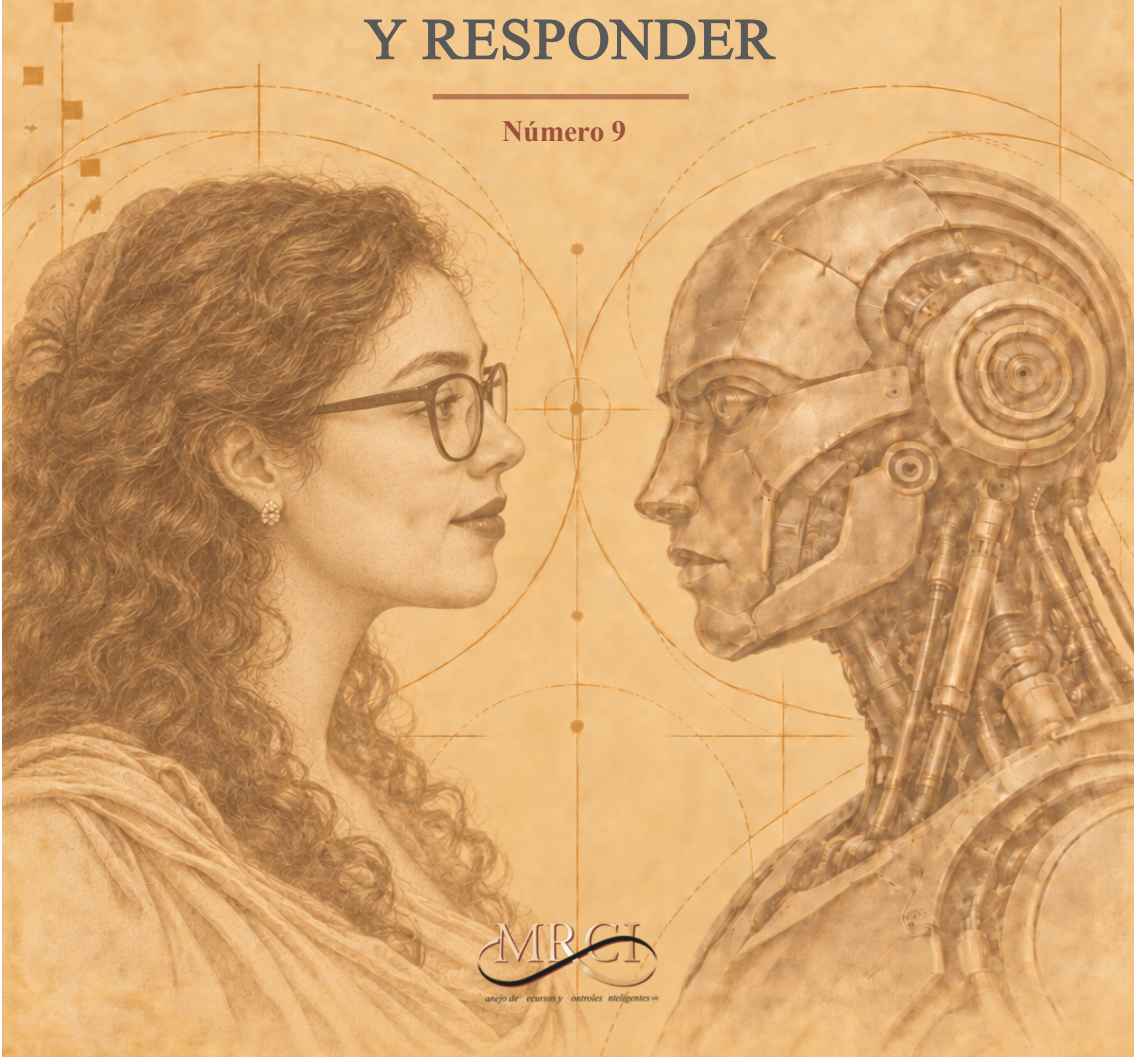
COLECCIÓN

MITOS Y REALIDADES DE LA IA

Entre la promesa y la realidad

ENTRENAMIENTO E INFERENCIA Y LA DIFERENCIA ENTRE AJUSTAR PARÁMETROS Y RESPONDER

Número 9



MRCI

anejo de ecursos y ontroles nteligentes



DR. JOSÉ ANTONIO PÉREZ RAMOS

ENTRENAMIENTO E INFERENCIA Y LA DIFERENCIA ENTRE AJUSTAR PARÁMETROS Y RESPONDER



SOBRE EL AUTOR

Doctor en Ciencias de lo Fiscal por el Instituto de Especialización para Ejecutivos. Maestro en Fiscal por la Universidad Autónoma "Benito Juárez" de Oaxaca. Maestro en Estudios Avanzados en Derechos Humanos por la Global Open University. Licenciado en Contaduría Pública por la Universidad Autónoma "Benito Juárez" de Oaxaca y Licenciado en Filosofía por la Universidad Nacionalista de México.

Socio fundador y director general de la Firma Manejo de Recursos y Controles Inteligentes (Firma MRCI). Fiscalista del año 2009 por la revista *Defensa Fiscal*. Doctor honoris causa por la Asamblea General del Claustro Doctoral Iberoamericano (2016). Doctor honoris causa por el Instituto Mexicano de Líderes de Excelencia (2017). Doctor honoris causa por 1Million Startups, Latinomics, Leaderships Forum y la Fundación Humanist World (2018). Doctor honoris causa por el Claustro de Doctores de la AISOL (American International School of Law) (2019).

Autor de diversas obras, entre las que destacan *Ius mutantur o del derecho que cambia* (Tirant Lo Blanch, 2025), *Futurología* (MRCI, 2025) y *Hasta que la muerte nos separe* (MRCI, 2025). Ha participado como coordinador en *Cuestiones tributarias. Problemas y controversias en el México actual* (Tirant Lo Blanch, 2023), *El costo de la justicia* (Apex Iuris, 2019), *Remuneraciones estratégicas inteligentes* (MRCI, 2015) y *Retos de la justicia tributaria* (Tirant Lo Blanch-AMDF, 2025).

Titular de *Pie de Página*, programa para la divulgación de los derechos humanos en México, a través de la plataforma Legal Cloud. Expositor en foros nacionales e internacionales. Colaborador en diversas revistas jurídicas, entre las que destacan *Defensa Fiscal* y *Legal Cloud*.

ENTRENAMIENTO E INFERENCIA Y LA DIFERENCIA ENTRE AJUSTAR PARÁMETROS Y RESPONDER

DR. JOSÉ ANTONIO PÉREZ RAMOS

PRIMERA EDICIÓN, ATEMPORAL

Derechos reservados, propiedad de
Dr. José Antonio Pérez Ramos

Comentarios y opiniones: investigacion@mrci.com.mx

Título original: Entrenamiento e inferencia y la diferencia entre ajustar parámetros y responder.

Colección mitos y realidades de la inteligencia artificial.

Entre la promesa y la realidad.

Autor: Dr. José Antonio Pérez Ramos.

Queda prohibida la reproducción total y parcial de esta obra denominada; ENTRENAMIENTO E INFERENCIA Y LA DIFERENCIA ENTRE AJUSTAR PARÁMETROS Y RESPONDER, por cualquier medio, sin autorización escrita del autor.

PRINTED IN MEXICO
IMPRESO EN MÉXICO

ÍNDICE

LA PREGUNTA DETRÁS DE UNA RESPUESTA MEJOR ..... - 7 -
MÉTODO Y FECHA DE CORTE ..... - 9 -
LOS OBJETOS QUE PUEDEN CAMBIAR ..... - 10 -
ENTRENAR CONSISTE EN EJECUTAR UN PROCEDIMIENTO DE AJUSTE ..... - 13 -
UN CÁLCULO MÍNIMO QUE HACE VISIBLE LA DIFERENCIA ..... - 16 -
PREENTRENAMIENTO Y AJUSTES POSTERIORES ..... - 19 -
AJUSTAR UNA PARTE TAMBIÉN ES ENTRENAMIENTO ..... - 22 -
INFERIR ES UTILIZAR UNA CONFIGURACIÓN PARA CALCULAR SALIDAS ..... - 23 -
LA INFERENCIA ADMITE DISTINTOS PROCEDIMIENTOS ..... - 27 -
LA CACHÉ AHORRA CÁLCULO Y REQUIERE SU PROPIA EXPLICACIÓN ..... - 29 -
APRENDER DEL CONTEXTO SIN ACTUALIZAR PESOS ..... - 31 -
DOCUMENTOS RECUPERADOS Y MEMORIA EXTERNA ..... - 33 -
LA EXCEPCIÓN REAL DE ADAPTAR DURANTE LA PRUEBA ..... - 35 -
LO QUE PERMITE DEMOSTRAR UNA CORRECCIÓN ..... - 37 -
ELEGIR UNA INTERVENCIÓN SEGÚN EL PROBLEMA ..... - 41 -
DATOS DE CONVERSACIÓN Y ENTRENAMIENTO POSTERIOR ..... - 43 -
OBJECIONES Y LÍMITES DE LA EXPLICACIÓN ..... - 44 -
CONCLUSIÓN ..... - 47 -
REFERENCIAS ..... - 48 -

La pregunta detrás de una respuesta mejor

Una persona corrige la fecha que aparece en una respuesta y el asistente la utiliza correctamente en el mensaje siguiente; la secuencia parece sencilla, pero admite explicaciones técnicas diferentes. La aplicación pudo añadir la corrección a la entrada del modelo, guardar un registro para recuperarlo después o consultar un documento nuevo; también podría formar parte de un sistema que ajusta parámetros, aunque esa posibilidad necesita evidencia específica.

El problema aparece cuando se utiliza el mismo verbo, aprender, para describir todos esos cambios y se extraen consecuencias que el mecanismo no justifica. Que el dato vuelva a aparecer mañana no demuestra que esté almacenado en los pesos del modelo; que desaparezca al abrir otra conversación tampoco demuestra que la primera corrección fuera inútil, porque pudo resolver correctamente la tarea local para la que se proporcionó.

La pregunta central de este artículo es qué cambia cuando una inteligencia artificial se entrena y qué cambia cuando responde. La tesis sostiene que una explicación suficiente debe identificar al menos el componente modificado, el



procedimiento que lo modifica, la duración prevista del cambio y la prueba utilizada para evaluar su efecto; observar únicamente el texto final deja abiertas varias de esas cuestiones.

Conviene distinguir además el modelo de la aplicación que lo utiliza. Una aplicación puede incorporar reglas, una base documental, servicios de búsqueda, instrucciones, filtros y una interfaz, por lo que una actualización del servicio no tiene que coincidir con una nueva versión del modelo; esta distinción permite investigar una mejora sin atribuir automáticamente todo el mérito a sus parámetros.

El alcance se concentra en modelos neuronales y, especialmente, en generación de lenguaje; no pretende convertir ese caso en una explicación exhaustiva de todo el aprendizaje automático. Existen procedimientos que aprenden árboles, conjuntos de reglas u otras estructuras, mientras el lenguaje de pesos, gradientes y redes corresponde a la familia técnica que aquí se examina.

Método y fecha de corte

Se realizó una revisión documental dirigida de publicaciones originales y documentación técnica, seleccionadas por su capacidad para separar mecanismos concretos; no es una revisión sistemática de toda la literatura ni un experimento sobre un servicio comercial. Las fuentes se consultaron para este artículo con corte al 2 de octubre de 2026, fecha local de Oaxaca, y se identifican sus versiones cuando la referencia corresponde a una prepublicación.

Los resultados de los trabajos citados se distinguen de las interpretaciones del artículo, de las recomendaciones de evaluación y de los casos hipotéticos. El cálculo de un modelo lineal se incluye únicamente como demostración matemática reproducible; no representa el entrenamiento de un modelo de lenguaje ni autoriza estimaciones sobre su precisión, costo o consumo de energía.

La exposición enlaza el ciclo de optimización con los estados intermedios de la inferencia y los problemas de mantenimiento de una aplicación; la memoria de productos específicos, el diseño completo de sistemas de recuperación y las políticas jurídicas del tratamiento de

datos quedan fuera de su alcance. Esta delimitación permite examinar la diferencia técnica sin convertirla en una recomendación universal de implementación.

Los objetos que pueden cambiar

Un parámetro es una cantidad ajustable que participa en la función del modelo; en una red neuronal, los pesos de sus transformaciones constituyen ejemplos centrales. Un conjunto de parámetros determina cómo procesa entradas, pero no equivale a una colección de respuestas escritas ni a una ficha independiente para cada hecho; su significado operativo depende de las operaciones y de los demás componentes con los que se combina.

Las activaciones son valores calculados mientras se procesa una entrada. Si dos preguntas producen activaciones diferentes, el modelo puede generar salidas distintas con los mismos pesos; por tanto, la presencia de movimiento numérico dentro de la red no basta para identificar entrenamiento, porque procesar información exige cambios transitorios aun cuando la configuración aprendida permanezca fija.

El contexto comprende la información que el modelo recibe para una ejecución, dentro de los límites del sistema; puede incluir instrucciones, mensajes anteriores, ejemplos y resultados de herramientas. La aplicación decide qué material conserva, selecciona o resume antes de enviarlo; el historial visible en una pantalla y la entrada efectiva del modelo no deben suponerse idénticos sin comprobar cómo se construye la solicitud. La memoria externa es almacenamiento gestionado fuera de los parámetros considerados: archivos, registros, preferencias o índices consultables. En este artículo se utiliza esa expresión en sentido funcional, sin atribuir experiencia subjetiva ni recuerdo humano al programa; un registro puede persistir durante mucho tiempo y, sin embargo, no ser recuperado en una determinada respuesta.

La Tabla 1 permite comparar esos objetos por su función, no por el tamaño que ocupan ni por el nombre comercial que una aplicación les asigne. Una cantidad persistente no tiene que ser un parámetro aprendido, mientras una cantidad ajustada mediante aprendizaje puede descartarse al terminar una tarea; persistencia y entrenamiento son dimensiones relacionadas, pero distintas.

Tabla 1. *Objetos que intervienen en un cambio de respuesta*

Objeto	Función en el proceso	Qué habría que comprobar
Parámetros aprendidos	Definen parte de las transformaciones del modelo	Si un procedimiento de ajuste modificó sus valores
Activaciones	Representan resultados intermedios de una ejecución	Con qué entrada y configuración se calcularon
Contexto	Proporciona información disponible para la respuesta	Qué instrucciones y materiales entraron realmente
Caché de cálculo	Reutiliza resultados compatibles ya calculados	Qué se conserva, cuándo se invalida y con qué alcance
Memoria externa	Guarda información que puede consultarse después	Qué registro se recuperó y cómo llegó a la entrada
Configuración del servicio	Organiza modelos, herramientas y reglas de uso	Qué versión o componente cambió entre consultas

Nota. Elaboración propia; las filas describen funciones que pueden coexistir en una aplicación y no categorías mutuamente excluyentes de productos.



Entrenar consiste en ejecutar un procedimiento de ajuste

Para comprender el entrenamiento conviene partir de una tarea concreta. Se dispone de datos, una función que produce resultados y un criterio para valorar esos resultados; el procedimiento modifica cantidades ajustables con la intención de mejorar ese criterio, aunque la intención de optimización no garantice una mejora en todos los usos posteriores.

En el aprendizaje supervisado, un ejemplo puede asociar una entrada con una salida de referencia; en el aprendizaje autosupervisado, la señal de entrenamiento se construye a partir de los propios datos. En un modelo de lenguaje autorregresivo, una forma de esa señal consiste en utilizar una parte de una secuencia para predecir el elemento siguiente, de manera que un corpus de texto permite construir muchas situaciones de predicción sin anotar manualmente cada continuación.

Bengio et al. (2003, pp. 1139–1143) presentan un antecedente original de modelado neuronal del lenguaje que aprende conjuntamente representaciones de palabras



y una función de probabilidad de secuencias. Su formulación permite observar la diferencia entre evaluar una probabilidad con parámetros dados y ajustar esos parámetros mediante un criterio de entrenamiento; el artículo histórico no describe los actuales servicios conversacionales completos ni pretende hacerlo.

Una función de pérdida traduce la discrepancia relevante a una cantidad que el algoritmo puede utilizar. En una tarea de clasificación, por ejemplo, se puede penalizar que el modelo asigne poca probabilidad a la clase de referencia; en una tarea de regresión, se puede medir la diferencia entre un valor predicho y uno observado, aunque elegir esa medida ya supone una decisión sobre qué errores importan.

El gradiente reúne derivadas que indican cómo cambia localmente la pérdida cuando cambian los parámetros. No constituye una explicación humana de por qué una respuesta es adecuada y tampoco representa una sentencia sobre la verdad de una frase; proporciona información numérica para un procedimiento de optimización definido, sujeto a los datos, al objetivo y a las condiciones de cálculo.



La documentación de PyTorch (2026, secciones «Optimizer» y «Full Implementation») distingue el cálculo de la pérdida, la propagación de sus derivadas y el paso del optimizador que ajusta parámetros. Esa separación ofrece una precaución importante: calcular una respuesta, una pérdida o incluso un gradiente no demuestra que se haya ejecutado una actualización; hay que identificar la operación que efectivamente cambió los valores y el componente al que se aplicó.

La tasa de aprendizaje controla la magnitud de ciertos pasos de ajuste; un lote reúne ejemplos utilizados en una operación o acumulación, y una época suele referirse a un recorrido por el conjunto de entrenamiento. Estos términos describen la organización del procedimiento, pero sus valores aislados no bastan para comparar calidad: también importan la distribución de datos, el objetivo, las restricciones del modelo y la evaluación posterior.

Un archivo que conserva un punto del entrenamiento, a menudo llamado checkpoint, permite retener una configuración y, según su contenido, reanudar el proceso. Para reproducir una trayectoria puede hacer falta guardar información adicional a los pesos, como el estado del

optimizador y la preparación de los datos; por ello, copiar únicamente un archivo de parámetros no equivale siempre a conservar todo lo necesario para repetir el entrenamiento.

Un cálculo mínimo que hace visible la diferencia

Consideremos un ejemplo hipotético con una entrada x, un único peso w y una predicción definida como w × x. El objetivo del ejercicio es mostrar qué operación cambia el peso; no se pretende que esta función reproduzca la complejidad de una red de lenguaje, sus millones de relaciones ni las dificultades de evaluar una respuesta abierta.

Se toma x = 2, un valor de referencia igual a 4 y un peso inicial w = 1; la predicción inicial es entonces 2. Definimos la pérdida como la mitad del cuadrado del error, $L = 0.5 \times (w \times x - 4)^2$, de modo que la pérdida inicial vale 2; todos estos valores pertenecen al escenario construido y no son mediciones de una aplicación real.

La derivada respecto de w es $(w \times x - 4) \times x$, que en el punto inicial vale −4. Con una tasa de aprendizaje de 0.1,

el paso de descenso se calcula restando al peso el producto de esa tasa y la derivada: w nuevo = 1 − 0.1 × (−4) = 1.4; aquí sí existe una actualización paramétrica explícita y puede compararse el valor anterior con el posterior.

Al volver a evaluar la misma entrada con el peso actualizado se obtiene 1.4 × 2 = 2.8 y una pérdida de 0.72. La pérdida disminuyó para este ejemplo y este paso, pero no llegó a cero; tampoco se examinó una población de datos, de modo que sería incorrecto convertir la operación en evidencia de exactitud general o afirmar que el procedimiento ya aprendió cualquier relación numérica.

La Tabla 2 mantiene separado el ajuste de las evaluaciones posteriores. Si se congela w en 1.4 y se introduce x = 3, la función produce 4.2 sin necesidad de modificar el peso; una salida nueva puede proceder simplemente de una entrada nueva, aun en un sistema determinista y completamente transparente.

Tabla 2. *Cálculo hipotético con un parámetro escalar*

Momento	Entrada x	Peso w	Predicción
Antes del ajuste	2	1	2
Después de un paso	2	1.4	2.8
Uso con entrada distinta	3	1.4	4.2

Nota. Elaboración propia; la referencia 4 y las pérdidas 2 y 0.72 corresponden únicamente a las dos primeras filas, no a la entrada distinta de la tercera fila.

Este cálculo permite además separar acierto y mecanismo. Cambiar manualmente el peso también modificaría la predicción, pero describir ese cambio como entrenamiento exigiría explicar el procedimiento que lo produjo; añadir una nota junto al resultado no cambiaría la función, aunque pudiera ayudar a una persona a interpretarlo correctamente.



La elección de la tasa tiene consecuencias visibles incluso aquí. Si se utilizara una tasa de 1 con los mismos valores iniciales, el nuevo peso sería 5, la predicción sería 10 y la pérdida ascendería a 18; por tanto, disponer de un gradiente no garantiza que cualquier paso disminuya la pérdida, y esta advertencia matemática no requiere extrapolar porcentajes a modelos reales.

Por último, una repetición de la entrada x = 2 con el peso fijo 1.4 produce nuevamente 2.8 en aritmética exacta. Esa propiedad pertenece al ejemplo determinista construido; no establece que todas las aplicaciones generativas deban contestar siempre con el mismo texto, ya que su procedimiento de selección, su contexto y otros componentes pueden variar sin entrenamiento adicional.

Preentrenamiento y ajustes posteriores

El preentrenamiento designa una fase inicial relativa a una trayectoria de desarrollo, generalmente destinada a aprender regularidades reutilizables antes de otras adaptaciones. La palabra no implica que el modelo haya visto todos los documentos, que conserve cada uno literalmente o que su información permanezca actualizada;

describe una posición dentro del proceso, no una garantía de cobertura.

El ajuste fino parte de una configuración previamente entrenada y continúa el aprendizaje para un objetivo o una distribución de uso. Puede modificar numerosos parámetros o solamente algunos; la diferencia práctica exige especificar qué componentes permanecieron congelados, qué ejemplos se utilizaron y cuál fue el criterio de selección de la versión obtenida, en vez de tratar cualquier personalización como ajuste fino.

El posentrenamiento es una denominación más amplia para intervenciones posteriores al entrenamiento inicial, cuyo contenido depende del trabajo o proveedor que la emplee. Puede incluir aprendizaje a partir de demostraciones o preferencias y otras técnicas; al interpretar una descripción técnica resulta más informativo identificar sus operaciones que suponer una receta universal por compartir el mismo nombre.

Ouyang et al. (2022, sección 3.1) documentan una secuencia concreta: ajuste supervisado con demostraciones, entrenamiento de un modelo de

recompensa a partir de comparaciones y optimización de una política utilizando esa recompensa. Es un caso histórico verificable de entrenamiento con retroalimentación humana; no prueba que pulsar un botón de valoración en cualquier producto modifique inmediatamente el modelo que está atendiendo la conversación.

La retroalimentación requiere una traducción entre la observación humana y el procedimiento de aprendizaje. Una corrección, una clasificación de respuestas o una señal de satisfacción pueden constituir insumos diferentes, con errores, sesgos y ambigüedades propios; reunirlos en un archivo todavía no establece cómo serán seleccionados, si serán utilizados ni cuándo se desplegará una configuración nueva.

La disponibilidad de una versión ajustada tampoco obliga a ponerla en producción. Es posible conservar varias candidatas, comparar resultados y descartar una modificación que mejora el comportamiento buscado pero deteriora un requisito importante; por tanto, el recorrido desde una interacción hasta un sistema actualizado puede



contener decisiones y evaluaciones que no son visibles en la interfaz.

Ajustar una parte también es entrenamiento

Hu et al. (2021, sección 4.1) proponen LoRA, que mantiene congelada una matriz de pesos preentrenada y aprende matrices adicionales cuya composición representa una actualización de bajo rango. El caso demuestra que conservar los pesos base no basta para afirmar ausencia de entrenamiento si existen componentes auxiliares entrenables; debe examinarse el conjunto que realmente participa en la función utilizada.

Un adaptador entrenado puede guardarse y combinarse con una base compatible, de modo que identificar una versión exige registrar ambos elementos. Esta consecuencia operativa no convierte cualquier archivo complementario en un adaptador: un documento de instrucciones conserva texto, mientras las matrices aprendidas forman parte de transformaciones numéricas; compartir la función general de especializar una aplicación no elimina esa diferencia.



También conviene separar entrenar el adaptador y cargar uno ya entrenado. La segunda operación cambia la configuración efectiva de la aplicación y puede alterar su conducta, aunque no ejecute en ese momento una nueva optimización; si un servicio alterna entre configuraciones, sus respuestas podrían cambiar sin que el usuario haya entrenado ninguna de ellas.

La técnica elegida tampoco resuelve por sí misma la pregunta sobre calidad. Una especialización puede favorecer el formato o la clasificación de cierto tipo de solicitudes y continuar fallando en casos nuevos; el beneficio debe definirse respecto de la tarea, sus datos y sus restricciones, sin convertir la eficiencia del ajuste en una promesa de verdad o de ausencia de errores.

Inferir es utilizar una configuración para calcular salida.

La inferencia es el uso del modelo para producir una predicción, una clasificación, una representación o una salida generada a partir de información disponible. En el caso ordinario de pesos fijos, el cálculo modifica estados intermedios sin actualizar los parámetros aprendidos; esa



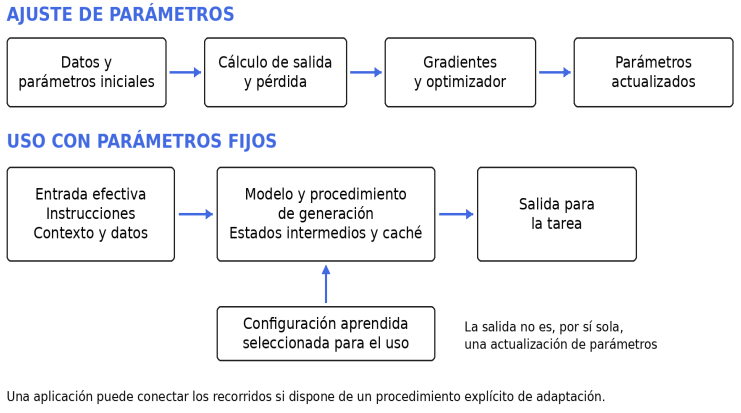
condición debe decirse expresamente para no borrar las modalidades que combinan uso y adaptación.

Responder no significa recuperar obligatoriamente una frase completa de un almacén. En un modelo autorregresivo de lenguaje, la distribución de una secuencia se organiza mediante probabilidades condicionadas por los elementos anteriores; esta estructura aparece en el planteamiento de Bengio et al. (2003, sección 2), aunque su modelo histórico y las implementaciones actuales difieran en muchos otros aspectos.

Un esquema simplificado de generación calcula valores para posibles continuaciones, aplica un procedimiento de selección y extiende la secuencia con lo seleccionado; esas unidades se denominan tokens y no tienen que coincidir con palabras completas. La siguiente operación utiliza una secuencia que ya incluye ese elemento, por lo que la entrada efectiva cambia mientras la respuesta avanza; esa dependencia permite entender cómo se desarrolla un texto sin imaginar una actualización del modelo después de cada unidad generada.

La Figura 1 separa dos recorridos funcionales. En el recorrido de ajuste, una señal de pérdida y un procedimiento de optimización producen parámetros nuevos; en el recorrido de uso, la configuración y la entrada producen salidas, con estados intermedios que pueden conservarse durante la ejecución; el esquema no impide que una aplicación conecte ambos recorridos cuando esa conexión está implementada y documentada.

Figura 1. *Dos recorridos que conviene identificar por separado*



Nota. Diagrama conceptual de elaboración propia; las flechas representan dependencias funcionales y no



tiempos de ejecución medidos ni una especificación de todos los sistemas.

En los generadores estudiados por Kwon et al. (2023, sección 2.2), el procesamiento del mensaje inicial y la generación autorregresiva presentan necesidades de cálculo diferentes. El primero aprovecha que los elementos de la entrada ya están disponibles; durante la generación se reutilizan claves y valores calculados previamente y se incorporan los correspondientes a los nuevos elementos, sin que eso suponga reentrenar los pesos.

Esas fases ayudan a interpretar la experiencia del usuario: esperar antes de ver el primer fragmento y observar después un flujo de texto pueden corresponder a trabajos distintos. Sin embargo, la latencia visible también depende de colas, red, herramientas y decisiones del servicio, por lo que no permite reconstruir de manera concluyente su implementación ni medir por sí sola la dificultad intelectual de una pregunta.

La inferencia admite distintos procedimientos

Que los parámetros permanezcan fijos no obliga a realizar exactamente el mismo trabajo computacional en cada solicitud. Una aplicación puede producir una sola propuesta, generar varias candidatas, solicitar verificaciones adicionales o utilizar herramientas antes de componer la salida; para describir su comportamiento se necesita considerar el procedimiento completo y el presupuesto disponible, además de la identidad del modelo.

Snell et al. (2024, secciones 2–3) estudian estrategias de cómputo durante la prueba, entre ellas revisión de propuestas y búsqueda guiada por verificadores, con modelos preparados para esas capacidades. Su análisis muestra que la utilidad de asignar más cálculo depende de la estrategia y de la dificultad de las preguntas estudiadas; no autoriza afirmar que un texto más largo o una espera mayor garanticen una respuesta correcta.

La palabra verificar merece una cautela adicional cuando el verificador también es un modelo. Una puntuación aprendida puede ser útil para seleccionar candidatos, pero debe evaluarse contra un criterio independiente adecuado



a la tarea; varias respuestas que se apoyan en el mismo supuesto equivocado no se convierten en evidencia sólida solamente porque coincidan entre sí.

Leviathan et al. (2023, sección 2 y apéndice A.1) presentan una decodificación especulativa que propone continuaciones con un modelo auxiliar y las comprueba mediante el modelo objetivo. En las condiciones de su algoritmo, el muestreo conserva la distribución objetivo mientras permite acelerar la generación; el ejemplo muestra que modificar la ejecución no implica necesariamente reentrenar el modelo ni alterar la distribución que se desea muestrear.

Esta propiedad no equivale a obtener siempre la misma cadena de texto en dos ejecuciones concretas. Igualdad de distribuciones, igualdad de resultados y coincidencia de los pasos internos son afirmaciones diferentes; al comparar dos procedimientos es necesario indicar cuál de ellas se estudia y con qué supuestos, sin trasladar una demostración particular a cualquier optimización de un servicio.

La caché ahorra cálculo y requiere su propia explicación

Una caché conserva resultados que pueden reutilizarse bajo condiciones de compatibilidad. En generación de lenguaje, la caché de claves y valores almacena representaciones calculadas para elementos anteriores, mientras las matrices que produjeron esas representaciones pertenecen a los parámetros del modelo; confundir ambos objetos convierte un mecanismo de ejecución en una supuesta modificación de conocimiento.

Kwon et al. (2023, secciones 2–4) analizan la gestión de esa memoria y proponen PagedAttention para reducir desperdicio y facilitar su uso compartido cuando corresponde. Su contribución concierne al servicio de inferencia y a sus recursos, no a un proceso que enseña hechos al modelo; la memoria ocupada por una solicitud no constituye, por su volumen, evidencia de aprendizaje persistente.

Un ejemplo hipotético permite fijar el alcance. Si un programa conserva los resultados compatibles del comienzo de una entrada extensa, puede evitar calcular otra vez esa misma parte al continuar el trabajo; cuando

cambian los elementos de los que depende el resultado conservado, debe existir una política correcta de invalidación o de reconstrucción, pues reutilizar valores incompatibles afectaría la ejecución.

También puede haber una caché de respuestas completas, que devuelve una salida previamente guardada cuando se cumplen criterios definidos por la aplicación. Es un mecanismo diferente de la caché de representaciones; obtener una respuesta muy rápida o idéntica a otra no identifica cuál se utilizó, y tampoco prueba que haya ocurrido una optimización de parámetros entre consultas.

El tiempo de conservación necesita describirse por separado. Una caché puede durar una ejecución o mantenerse para solicitudes compatibles, mientras una preferencia externa puede conservarse durante meses; la duración depende del diseño y de las políticas del sistema, de modo que no conviene asignar automáticamente a toda caché la condición de recuerdo personal ni a toda memoria la función de acelerar cálculo.



Aprender del contexto sin actualizar pesos

Brown et al. (2020, sección 2) distinguen el ajuste fino de la presentación de instrucciones y ejemplos durante la inferencia sin actualizar pesos. En el segundo caso, denominado aprendizaje en contexto, los ejemplos forman parte del condicionamiento de la respuesta; la denominación reconoce una capacidad de adaptación observable sin afirmar que se haya ejecutado el mismo procedimiento que modifica los parámetros.

La diferencia puede verse con una tarea hipotética de clasificación. En una conversación se establece que «A» significa entrega recibida y «B» significa entrega pendiente; el modelo dispone ahora de esas convenciones para interpretar casos posteriores, aunque las letras tengan otros significados en otros entornos y no se haya entrenado una versión nueva para toda la organización.

Si se cambian deliberadamente las convenciones en una segunda entrada, una respuesta adecuada debería atender a las definiciones pertinentes para esa segunda tarea. La posibilidad de alternar instrucciones explica el valor de la adaptación contextual, pero también su alcance: el comportamiento observado corresponde a una



entrada y no demuestra que la convención se haya incorporado de manera permanente a todos los usos del modelo.

La desaparición de un dato puede tener varias causas. El material pudo no enviarse nuevamente, quedar fuera de la selección de contexto, perderse en un resumen o competir con instrucciones contradictorias; sin acceso a la entrada efectiva, culpar inmediatamente a un supuesto olvido de pesos confunde una hipótesis con un diagnóstico demostrado.

Tampoco debe interpretarse todo aprendizaje en contexto como una mera copia literal. Un sistema puede aplicar una regularidad a ejemplos diferentes de los mostrados; lo que necesita evaluación es la extensión y confiabilidad de esa aplicación, mientras el mecanismo operativo descrito sigue siendo distinto de una actualización paramétrica por gradientes.

La cautela tiene dos direcciones. Negar cualquier capacidad de adaptación porque los pesos están fijos empobrece la descripción de lo que el sistema hace; llamar reentrenamiento a cada ejemplo introducido oculta qué

información necesita seguir presente para sostener ese comportamiento y qué ocurrirá al cambiar la tarea.

Documentos recuperados y memoria externa

Lewis (2021, secciones 2.2–2.4) distinguen la memoria paramétrica del generador y un índice documental consultable en su formulación de generación aumentada por recuperación. Además, el trabajo entrena componentes del recuperador y del generador, por lo que sería incorrecto resumirlo diciendo que la recuperación nunca requiere entrenamiento; lo decisivo es separar el aprendizaje de esos componentes y la consulta posterior de documentos.

En el mismo estudio se examina el cambio del índice utilizado durante la prueba (Lewis et al., 2021, sección 4.5). La observación respalda una posibilidad concreta: cambiar información accesible externamente puede modificar respuestas sin volver a entrenar el generador en esa operación; no demuestra que añadir cualquier archivo a cualquier aplicación baste para que sea seleccionado y utilizado correctamente.



Imaginemos una organización que sustituye su manual de atención por una edición nueva. Para comprobar la actualización convendría elegir preguntas cuya respuesta realmente difiera entre las ediciones, identificar los fragmentos recuperados y revisar la salida; si ambas versiones conviven sin una regla clara de prioridad, la aplicación puede utilizar material obsoleto o combinar instrucciones incompatibles.

Este caso es hipotético y no informa resultados de una implantación real. Su valor está en localizar el problema: si la edición correcta nunca llegó a la entrada, entrenar el generador no aborda directamente esa falla; si llegó y fue interpretada mal, habrá que examinar la tarea, las instrucciones y el comportamiento del modelo, evitando atribuir causas a partir del resultado final solamente.

La incorporación documental puede contener operaciones distintas. Guardar un archivo, extraer su texto, dividirlo en fragmentos, calcular representaciones y actualizar un índice son pasos que conviene registrar individualmente; utilizar un modelo fijo para calcular representaciones es inferencia de ese modelo, aunque la colección resultante crezca y se haga más útil para el sistema.



Una preferencia guardada presenta una distinción similar. Si la aplicación almacena que una persona prefiere respuestas breves y recupera ese registro en la siguiente sesión, la persistencia observada puede proceder del almacenamiento externo; la frase «ahora lo recuerdo» describe una experiencia de uso, pero no proporciona una inspección de los parámetros ni acredita que estos hayan cambiado.

Guardar también requiere decidir qué se considera vigente. Una preferencia temporal, una corrección de hecho y una instrucción limitada a un proyecto no deberían convertirse por accidente en reglas universales; esta es una recomendación de diseño y evaluación, no una afirmación sobre cómo funcionan todas las aplicaciones existentes.

La excepción real de adaptar durante la prueba

La distinción anterior no permite concluir que ningún sistema entrena mientras se utiliza. Sun et al. (2020, sección 2) proponen un procedimiento de entrenamiento durante la prueba que utiliza una tarea autosupervisada para modificar parámetros compartidos antes de una predicción; sus experimentos se centran en clasificación

de imágenes ante cambios de distribución, y no describen una propiedad general de los asistentes de lenguaje.

El trabajo distingue una modalidad que descarta el ajuste específico después de la predicción y otra en línea que conserva el estado para la siguiente muestra. Esa diferencia es especialmente útil aquí: puede haber una actualización paramétrica real sin persistencia entre casos, y puede haber persistencia externa sin actualización paramétrica; ninguna de las dos dimensiones sustituye a la otra.

Por eso resulta más preciso describir un sistema mediante operaciones que mediante una frontera cronológica rígida. Un proceso puede alternar predicción, obtención de información, ajuste y nueva predicción; para afirmar qué ocurre en un producto concreto se necesitan documentación o registros de esa implementación, sin generalizar desde una investigación que demuestra una posibilidad técnica.

El aprendizaje en línea también introduce preguntas de control. Si el estado se conserva entre observaciones, el orden de las entradas puede convertirse en una variable



relevante para reproducir la ejecución; una evaluación debería informar la secuencia, el punto de inicio y la política de reinicio, en lugar de presentar cada respuesta como si hubiera surgido de una configuración independiente e invariable.

Mayor cómputo durante la prueba y entrenamiento durante la prueba no son expresiones intercambiables. Se puede invertir más cálculo en búsqueda o selección con componentes fijos, y se puede invertir cálculo en optimización que modifica componentes aprendidos; examinar el objeto del cálculo permite distinguirlos incluso cuando ambos procedimientos se realizan antes de entregar una respuesta.

Lo que permite demostrar una corrección

Retomemos el caso hipotético de una fecha equivocada. Si la persona proporciona la fecha correcta y la respuesta siguiente la incorpora, se ha observado una mejora local condicionada por nueva información; ese resultado puede ser útil por sí mismo, siempre que se compruebe el documento final en vez de conformarse con que el sistema afirme haber entendido.

Para investigar la persistencia, se podría volver a plantear la tarea en otra conversación y registrar la configuración disponible. Si la fecha aparece correctamente, la observación justifica investigar su procedencia: pudo llegar desde memoria externa, recuperación, instrucciones reutilizadas o un modelo distinto; la persistencia conductual aumenta lo que se sabe sobre el sistema, pero no identifica automáticamente una actualización de pesos.

La Tabla 3 propone un diseño de diagnóstico, todavía no ejecutado, que separa preguntas sobre resultado y mecanismo. Las comparaciones deben respetar las opciones realmente disponibles y no tratar una sesión nueva como garantía absoluta de ausencia de información previa; si la aplicación incorpora preferencias compartidas o datos del proyecto, seguirá existiendo contexto pertinente.

Tabla 3. *Preguntas y evidencia para investigar una corrección*

Pregunta	Observación o registro útil	Límite de la conclusión
¿Mejoró la respuesta inmediata?	Comparar la salida anterior y la posterior con la fuente correcta	No demuestra persistencia ni cambio de parámetros
¿Se mantiene fuera del diálogo?	Repetir con entrada y estado externo documentados	Un resultado estable no revela por sí solo su mecanismo
¿Se recuperó un registro?	Identificar el dato externo y su incorporación efectiva	Guardarlo no demuestra que se haya utilizado
¿Se ajustaron componentes?	Registrar procedimiento, versiones y componentes entrenables	Un archivo distinto no demuestra mejor desempeño
¿Se generalizó la corrección?	Evaluar casos nuevos y contraejemplos pertinentes	El alcance depende del conjunto examinado

Nota. Propuesta de evaluación de elaboración propia; no representa una prueba realizada ni una certificación técnica de un proveedor.

La generalización requiere casos donde la nueva información sea pertinente y casos donde no lo sea. Si se corrigió la fecha de un evento, no tendría sentido premiar que el sistema aplique esa fecha a cualquier evento mencionado; una evaluación que solo repite el ejemplo corregido puede confundir dependencia contextual con una distinción bien aprendida.

También hace falta registrar qué permaneció igual. Si entre dos consultas cambian modelo, instrucciones, repositorio y herramientas, una mejora real no permite atribuir el efecto a una intervención aislada; la explicación debe reconocer esos cambios conjuntos o utilizar comparaciones que controlen las variables relevantes cuando el entorno permita hacerlo.

Elegir una intervención según el problema

Una organización hipotética desea que su asistente clasifique solicitudes de mantenimiento con categorías internas y redacte una explicación breve. Antes de solicitar entrenamiento, conviene separar errores de información, errores de formato y errores de decisión; cada grupo sugiere pruebas diferentes y podría requerir modificaciones en componentes distintos.

Si las categorías están mal definidas, ningún ajuste compensa de manera fiable una contradicción que la organización no ha resuelto. El primer trabajo sería precisar criterios y ejemplos de frontera, porque etiquetar de forma inconsistente los casos trasladaría el desacuerdo al conjunto utilizado para evaluar o entrenar; esta observación es una consecuencia lógica del objetivo planteado, no un resultado experimental atribuido a una empresa.

Si el criterio es claro pero no está disponible en la entrada, incorporar instrucciones o ejemplos puede constituir una intervención inicial verificable. Si depende de un manual que cambia con frecuencia, convendría comprobar la consulta y el control de versiones; si persiste un patrón de

errores sobre una tarea estable y hay datos pertinentes, podría estudiarse una adaptación paramétrica y compararla con las alternativas.

La comparación debería utilizar una misma definición de éxito. Para esta tarea hipotética podrían separarse exactitud de la categoría, cumplimiento del formato, identificación de casos insuficientemente descritos y tiempo necesario para corregir el resultado; elegir únicamente la apariencia de seguridad de la explicación dejaría sin medir la decisión que la organización necesita.

Los datos reservados para evaluar no deberían utilizarse repetidamente para seleccionar instrucciones o ajustar componentes sin reconocerlo. A medida que las decisiones se adaptan a esos casos, dejan de representar con la misma independencia una prueba final; una propuesta razonable consiste en conservar un conjunto separado para la evaluación de cierre y documentar cualquier acceso previo.

La intervención debe incluir una ruta de reversión. Mantener una configuración anterior y versiones identificables permite comparar y retirar un cambio



perjudicial; no garantiza que el retroceso sea trivial en todos los sistemas, sobre todo cuando también cambiaron repositorios o estados externos, pero obliga a pensar qué habría que restaurar para reconstruir una ejecución.

Datos de conversación y entrenamiento posterior

Una conversación puede generar registros, correcciones o valoraciones que eventualmente formen parte de un proceso de mejora; esa posibilidad no equivale a afirmar que el modelo se actualiza con cada mensaje. Recogida, conservación, selección de datos, preparación del conjunto, optimización y despliegue son operaciones diferentes, y cada una necesita evidencia propia cuando se describe un servicio identificado.

Este artículo no evalúa la política de tratamiento de datos de ningún proveedor concreto. Esas condiciones pueden variar por producto, tipo de cuenta, contrato, configuración y fecha; por ello, una decisión sobre información confidencial requiere consultar las condiciones oficiales aplicables, en vez de deducirlas de una explicación general sobre inferencia con pesos fijos.

La distinción evita dos conclusiones erróneas. Ausencia de entrenamiento inmediato no significa ausencia de almacenamiento o de acceso a los datos; existencia de una opción para contribuir a mejoras tampoco identifica cuándo se usará una conversación específica ni demuestra que ya forme parte de una versión desplegada.

También importa el alcance de una eliminación. Retirar un documento de un repositorio consultable y modificar un modelo que hubiese sido entrenado con información relacionada son intervenciones distintas; comprobar la primera operación no permite certificar la segunda, por lo que cualquier promesa sobre borrado debe señalar el componente y la evidencia que realmente cubre.

Objeciones y límites de la explicación

Una primera objeción sostiene que, para el usuario, basta con que la aplicación mejore y resulta innecesario conocer el mecanismo. Eso puede ser suficiente para una tarea de bajo riesgo y alcance inmediato; cuando se espera persistencia, confidencialidad, reproducción o mantenimiento, la distinción determina qué se debe conservar, revisar o contratar y deja de ser una precisión meramente terminológica.



Otra objeción señala que una misma conducta puede tener explicaciones internas distintas incluso después de varias pruebas. Es correcto: las observaciones de una interfaz pueden descartar hipótesis simples o identificar dependencias, pero no siempre permiten demostrar el mecanismo; cuando no hay acceso a registros técnicos, corresponde separar la declaración del proveedor de lo verificado por la organización.

Un tercer límite concierne al vocabulario. Parámetros fijos suele significar que no se optimizan los parámetros considerados durante la operación descrita; no implica que todos los datos internos del servicio sean invariables ni que cualquier componente auxiliar permanezca idéntico, por lo que una afirmación rigurosa debe precisar el nivel al que se refiere.

La explicación tampoco resuelve el problema general de interpretar qué información representa cada peso. Saber que un procedimiento modificó un conjunto de parámetros no permite localizar de manera simple una creencia o garantizar que un dato específico se use correctamente; la prueba del mecanismo y la prueba del desempeño responden a preguntas diferentes.



Finalmente, las publicaciones citadas pertenecen a configuraciones y fechas determinadas. Se utilizan para explicar mecanismos demostrables y sus límites, no para declarar qué arquitectura utiliza hoy un producto cerrado, ni para trasladar a toda aplicación cifras de rendimiento obtenidas bajo condiciones que aquí no se han reproducido.



Conclusión

La diferencia entre entrenamiento e inferencia se vuelve clara cuando se identifica qué operación cambia qué componente. Entrenar ajusta cantidades aprendidas mediante un procedimiento definido; inferir utiliza una configuración para producir salidas, mientras el contexto, los estados de cálculo y el almacenamiento externo pueden modificar el resultado sin que se actualicen los pesos considerados.

La separación debe conservar sus excepciones. Existen ajustes parciales y procedimientos que entrenan durante la prueba, así como formas de inferencia que utilizan varios modelos, verificaciones o cálculos adicionales; ninguna de esas posibilidades justifica atribuirlas a una aplicación cuya implementación no se ha comprobado.

Para actuar con criterio, una corrección debe evaluarse primero por el resultado que se necesita y después por las afirmaciones adicionales que se quieran sostener. Mejorar una respuesta, conservar un dato, actualizar una biblioteca, cargar un adaptador y entrenar una configuración son logros diferentes; documentarlos con su

alcance permite elegir la intervención adecuada y reconocer qué sigue sin saberse.

Referencias

Bengio, Y., Ducharme, R., Vincent, P., y Jauvin, C. (2003). A neural probabilistic language model [Un modelo probabilístico neuronal del lenguaje]. *Journal of Machine Learning Research, 3*, 1137–1155. https://www.jmlr.org/papers/v3/bengio03a.html

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). *Language models are few-shot learners* [Los modelos de lenguaje aprenden con pocos ejemplos] (Versión 4) [Prepublicación]. arXiv. https://doi.org/10.48550/arXiv.2005.14165

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., y Chen, W. (2021). *LoRA: Low-rank adaptation of large language models* [LoRA: Adaptación de bajo rango de grandes modelos de lenguaje] (Versión 2) [Prepublicación]. arXiv. https://doi.org/10.48550/arXiv.2106.09685

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., y Stoica, I. (2023). *Efficient memory management for large language model serving with PagedAttention* [Gestión eficiente de memoria para el servicio de grandes modelos de lenguaje mediante PagedAttention] (Versión 1) [Prepublicación]. arXiv. https://doi.org/10.48550/arXiv.2309.06180

Leviathan, Y., Kalman, M., y Matias, Y. (2023). Fast inference from transformers via speculative decoding [Inferencia rápida de transformers mediante decodificación especulativa]. En A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, y J. Scarlett (Eds.), *Proceedings of the 40th International Conference on Machine Learning* (Vol. 202, pp. 19274–19286). PMLR. https://proceedings.mlr.press/v202/leviathan23a.html

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., y Kiela, D. (2021). *Retrieval-augmented generation for knowledge-intensive NLP tasks* [Generación aumentada por recuperación para tareas de procesamiento del lenguaje natural que requieren conocimiento] (Versión 4) [Prepublicación]. arXiv. https://doi.org/10.48550/arXiv.2005.11401

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., y Lowe, R. (2022). *Training language models to follow instructions with human feedback* [Entrenamiento de modelos de lenguaje para seguir instrucciones con retroalimentación humana] (Versión 1) [Prepublicación]. arXiv. https://doi.org/10.48550/arXiv.2203.02155

PyTorch. (2026, 31 de julio). *Optimizing model parameters* [Optimización de los parámetros del modelo]. Recuperado el 2 de octubre de 2026, de https://docs.pytorch.org/tutorials/beginner/basics/optimization_tutorial.html

Snell, C., Lee, J., Xu, K., y Kumar, A. (2024). *Scaling LLM test-time compute optimally can be more effective than scaling model parameters* [Escalar de forma óptima el cómputo de un modelo de lenguaje durante la prueba puede ser más eficaz que aumentar sus parámetros] (Versión 1) [Prepublicación]. arXiv. https://doi.org/10.48550/arXiv.2408.03314

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A. A., y Hardt, M. (2020). Test-time training with self-supervision for generalization under distribution shifts [Entrenamiento durante la prueba con autosupervisión para generalizar ante cambios de distribución]. En H. Daumé III y A. Singh (Eds.), *Proceedings of the 37th International Conference on Machine Learning* (Vol. 119, pp. 9229–9248). PMLR. https://proceedings.mlr.press/v119/sun20b.html

