RAZONAMIENTO EN GRANDES MODELOS DE LENGUAJE, UN MINI-ESTUDIO

Jorge A. Pichardo Cabrera, Franco Hernández Piloto, Hivan Cañizares Diaz, Alejandro García González, Ernesto Bárcena Trujillo, Carlos Mauricio Reyes Escudero
UH, MATCOM, CC, C411

<thinking>
El usuario solicita un estudio que se centre en dos aspectos principales relacionados con el razonamiento en Grandes Modelos de Lenguaje (LLMs):

1. Revisión exhaustiva de las diversas técnicas existentes para potenciar el razonamiento en LLMs.
2. Análisis de datasets relevantes utilizados para la evaluación del razonamiento en LLMs. Se examinará la filosofía de diseño detrás de cada dataset, los tipos de razonamiento que buscan medir, y las razones por las cuales representan un desafío para los modelos actuales.

Debo considerar la estructura de la redacción, dado que la intención del documento es servir como fuente directa de conocimiento de diversas investigaciones llevadas a cabo en el área.
</thinking>

Introducción

Este trabajo busca compilar las principales ideas y conceptos sobre el razonamiento en grandes modelos de lenguaje.
A continuación, presentamos un recorrido exploratorio sobre las principales técnicas y conjuntos de datos usados para evaluar el razonamiento en LLMs, en cada caso, considerando la idea central, y mostrando ejemplos gráficos donde sea necesario (con la citación correspondiente en cada caso).

Tenemos la intención de que compile ideas críticas, y sirva de apoyo para el lector empaparse en cara a su introducción en el campo.

Técnicas

scratchpad [Nye et al., 2021]: Los grandes modelos de lenguaje (LLMs) son buenos en tareas sencillas de 'una sola pasada', pero fallan en tareas que requieren múltiples pasos de cálculo, como sumar números grandes o ejecutar programas complejos. Sin embargo, si se les pide que hagan esas tareas 'paso a paso', mostrando los resultados de cada paso intermedio en una especie de 'hoja de borrador' (scratchpad), los LLMs pueden realizar esos cálculos complejos de manera mucho más efectiva, incluso con pocos ejemplos.

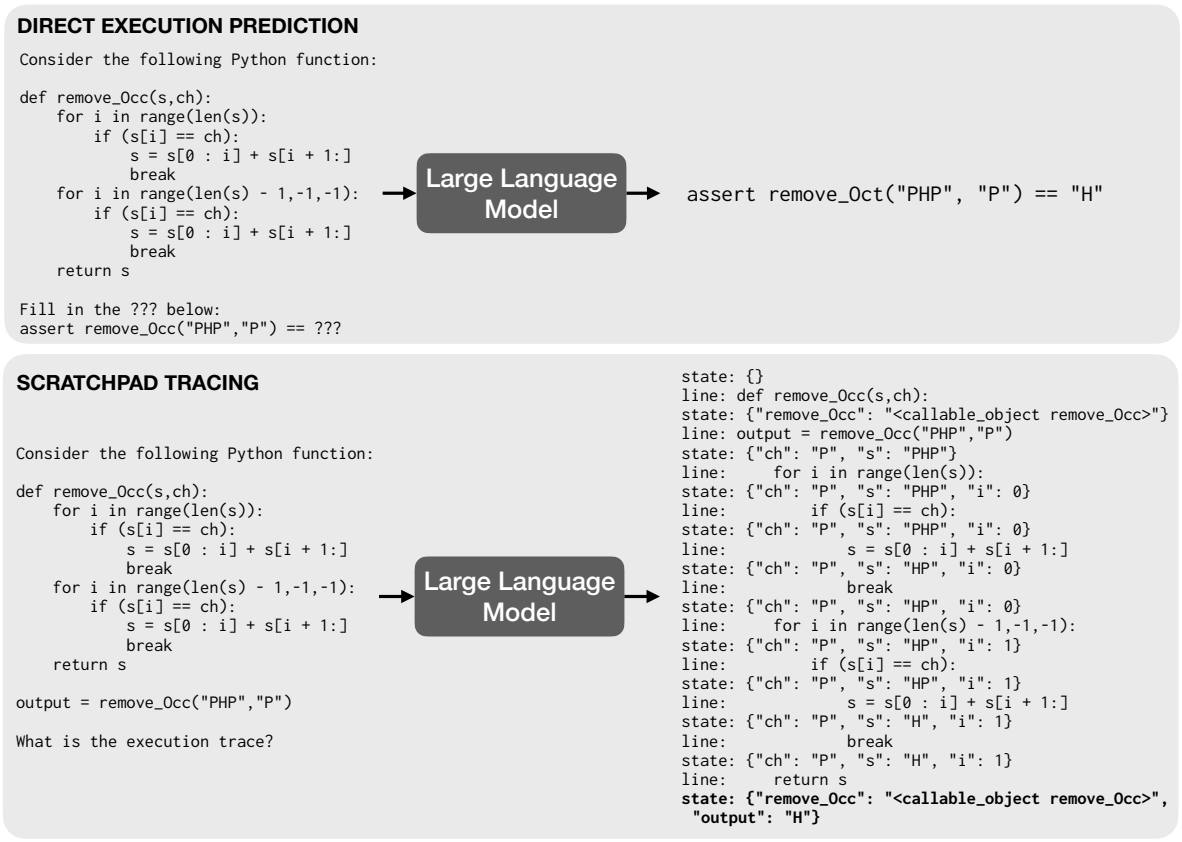

Figura 1: *Ejemplo de aplicación de scratchpad a tarea de predicción de ejecución de código. Figura tomada de [Nye et al., 2021]*

Chain of Thought(CoT) [Wei et al., 2023]: La clave para mejorar el razonamiento complejo en grandes modelos de lenguaje (LLMs) es guiarlos para que generen una 'cadena de pensamiento' (chain of thought). Esto significa que, en lugar de solo dar la respuesta final, el modelo debe producir una serie de pasos intermedios de razonamiento. Esta capacidad surge de forma natural en LLMs lo suficientemente grandes utilizando una técnica llamada 'chain of thought prompting', donde se les muestran unos pocos ejemplos de cómo generar estas cadenas de pensamiento en los prompts.

Muchos de los consecutivos acercamientos incluyen

mejoras a esta idea inicial.

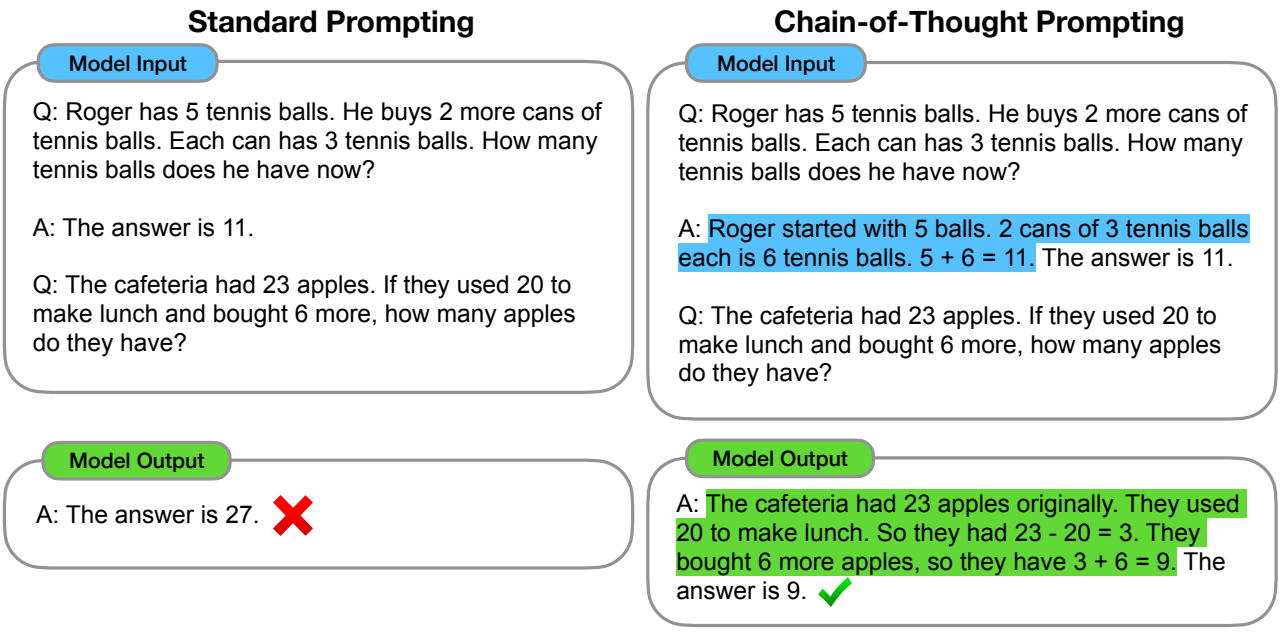

Figura 2: *CoT en tarea de razonamiento matemático. Figura tomada de [Wei et al., 2023]*

Zero-Shot CoT [Kojima et al., 2023]: Los grandes modelos de lenguaje (LLMs) no solo son buenos aprendiendo con pocos ejemplos (few-shot), como se creía, sino que también son capaces de razonar de manera efectiva sin ejemplos, es decir, 'zero-shot', con una sencilla modificación en el prompt. La clave es agregar la frase 'Pensemos paso a paso' antes de que el modelo genere la respuesta. Esta técnica, llamada 'Zero-shot-CoT' (Zero-shot Chain of Thought), hace que el LLM genere una cadena de pensamiento, es decir, que explicite los pasos intermedios del razonamiento, lo cual mejora drásticamente su rendimiento en tareas complejas.

Complexity-Based Prompting [Fu et al., 2023]: Partiendo de la base de que el 'Chain of Thought' (CoT) mejora el razonamiento en LLMs al generar pasos intermedios, el 'prompting basado en complejidad' propone un refinamiento: seleccionar ejemplos CoT para el prompt y respuestas generadas por el modelo, no solo por su validez, sino por la cantidad de pasos de razonamiento. Al elegir ejemplos más complejos para el prompt (es decir, con más pasos), y al favorecer las respuestas provenientes de cadenas de pensamiento más elaboradas, esta técnica guía al modelo a descomponer los problemas de manera más profunda y a generar soluciones más robustas. Esto se realiza tanto en el proceso de prompting, seleccionando los ejemplos del prompt, como en la decodificación, favoreciendo las respuestas que provienen de razonamientos más complejos, mostrando un aumento de la efectividad del LLM en tareas de razonamiento.

Self-Ask [Press et al., 2023]: Propone un método que supera el 'chain of thought' (CoT). En 'self-ask', el modelo se hace preguntas de seguimiento (y las responde) antes de responder a la pregunta inicial. Además, se demuestra que este tipo de prompting estructurado permite integrar un motor de búsqueda para responder a las preguntas de seguimiento, lo que mejora aún más la precisión.

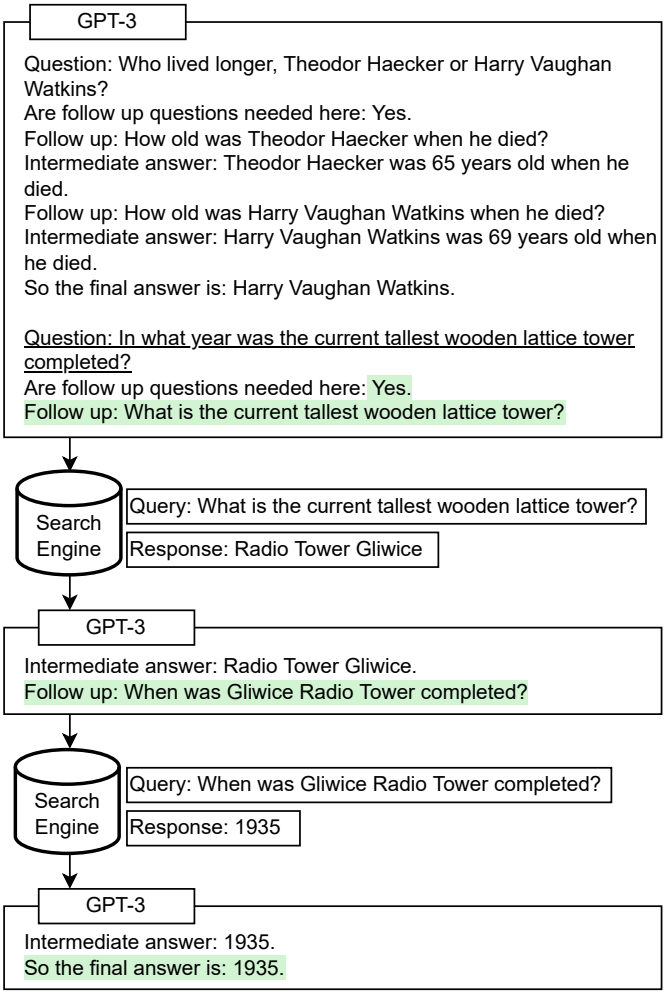

Figura 3: *Flujo de Self-Ask, tomado de [Press et al., 2023]*

Self-Verification [Weng et al., 2023]: En un avance reciente sobre las técnicas de razonamiento en grandes modelos de lenguaje (LLMs), se ha explorado la capacidad de los modelos para la 'auto-verificación'. Reconociendo que, aunque el 'chain of thought' (CoT) facilita el

razonamiento, sigue siendo propenso a errores, se ha propuesto un método donde el LLM revisa sus propias respuestas. Este proceso toma la conclusión obtenida mediante CoT y la usa como una condición adicional del problema original. Luego, realizando una verificación 'hacia atrás', el modelo asigna una puntuación a la validez de cada respuesta, seleccionando aquella que obtiene la puntuación más alta, logrando así mejorar la precisión en tareas complejas de razonamiento.

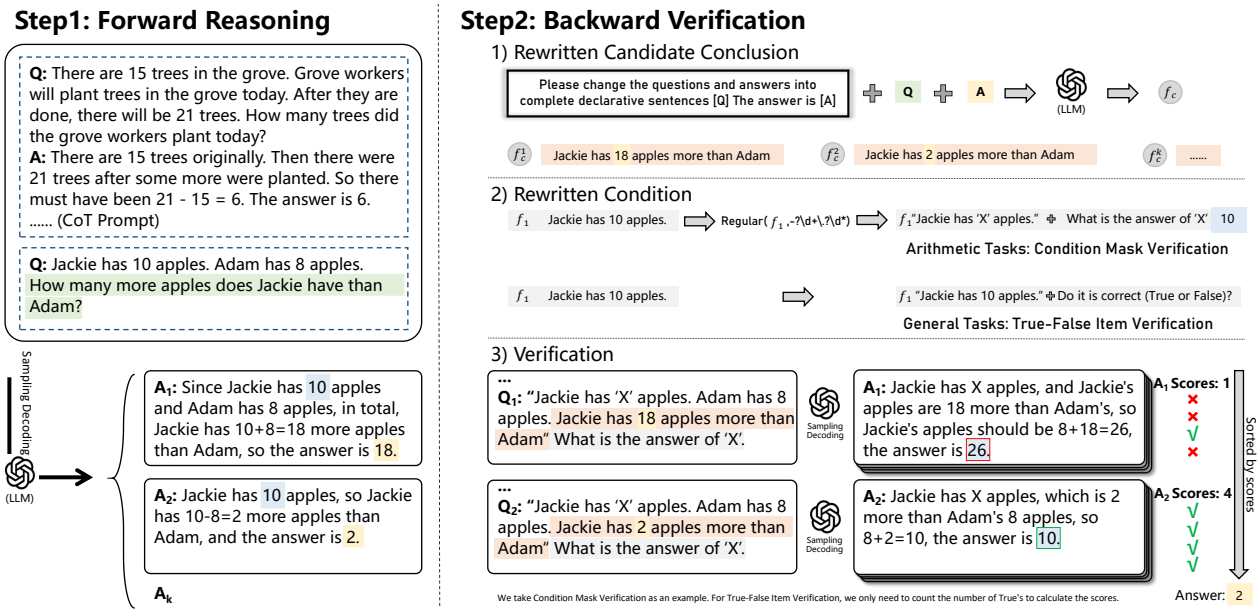

Figura 4: *Self-Verification, de [Weng et al., 2023]*

Self-Consistency [Wang et al., 2023]: Es una nueva estrategia de decodificación para mejorar el CoT en LLMs. En lugar de utilizar la decodificación 'greedy' tradicional, que solo toma la ruta de razonamiento más probable, la auto-consistencia primero muestrea un conjunto diverso de posibles rutas de razonamiento. Luego, selecciona la respuesta más consistente, evaluando el acuerdo de las respuestas obtenidas a través de las distintas rutas muestreadas. La idea es que un problema complejo de razonamiento suele admitir múltiples maneras válidas de pensarlo que llevan a una única respuesta correcta.

Self-Debugging [Chen et al., 2023a]: El Auto-Depurado es una técnica que capacita a los grandes modelos de lenguaje (LLMs) para depurar autónomamente su código mediante un proceso de auto-análisis 'few-shot'. Al igual que un programador que usa la técnica del 'rubber duck debugging' explicándose a sí mismo el código para encontrar errores, el LLM, mediante el examen de los resultados de ejecución y la auto-explicación del código en lenguaje natural, puede identificar y corregir sus errores sin necesidad de retroalimentación humana o mensajes de error.

Program of Thought(PoT) [Chen et al., 2023b]: Es una técnica que aprovecha modelos de lenguaje, especialmente Codex, para formalizar el proceso de razonamiento como un programa de código ejecutable. A diferencia de enfoques donde el modelo realiza directamente los cálculos y operaciones, PoT delega la computación a un sistema externo, como un intérprete o un entorno de ejecución. El LLM se enfoca en la generación de un programa que describe la lógica de razonamiento y la serie de operaciones necesarias para resolver un problema, pero no ejecuta directamente el programa. En cambio, un ordenador externo es el encargado de procesar y ejecutar este programa generado por el modelo para derivar la respuesta final. En esencia, el LLM se convierte en un diseñador de procesos de computación, formalizando el método de resolución como un programa de código, mientras que la computación en sí misma se externaliza a un entorno más apropiado para su ejecución precisa (ver también Program-aided-language (PAL) [Gao et al., 2023]).

REFINER [Paul et al., 2024]: Con el fin de mejorar su razonamiento mediante la generación explícita de inferencias intermedias, como en el 'chain-of-thought' (CoT) y reconociendo que estas inferencias intermedias pueden ser incorrectas, REFINER introduce un 'modelo crítico' que proporciona retroalimentación automatizada sobre el razonamiento. En concreto, el modelo crítico ofrece una retroalimentación estructurada que el modelo de razonamiento utiliza para mejorar iterativamente sus argumentos intermedios.

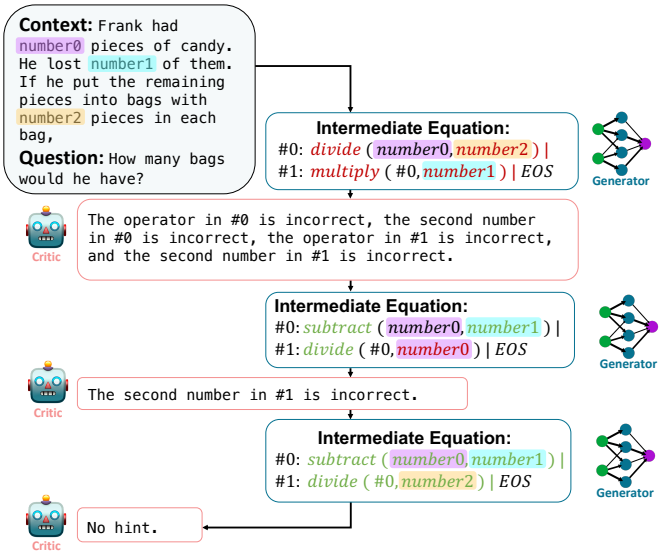

Figura 5: *REFINER, de [Paul et al., 2024]*

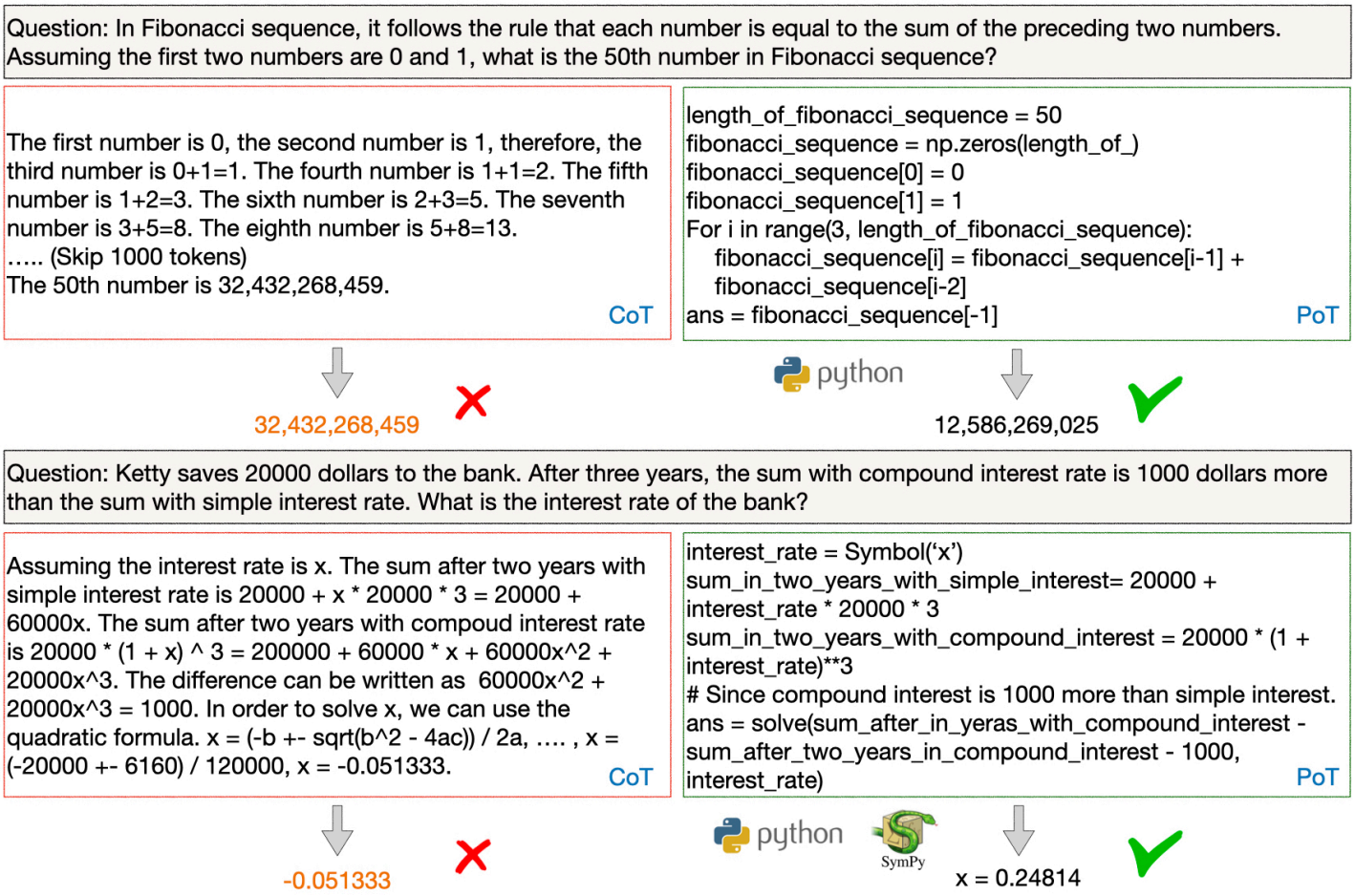

Figura 6: *PoT and PAL, de [Chen et al., 2023b]*

Self-Improvement [Huang et al., 2022]: Este trabajo demuestra que un LLM puede auto-mejorarse utilizando únicamente conjuntos de datos sin etiquetar. Se usa un LLM pre-entrenado para generar respuestas 'de alta confianza' con razonamiento aumentado para preguntas sin etiquetar, utilizando 'chain-of-thought' (CoT) y auto-consistencia. Luego, se ajusta el LLM utilizando estas soluciones autogeneradas como salidas objetivo. Este enfoque de auto-aprendizaje con datos sin etiquetar mejora las capacidades de razonamiento del LLM y permite alcanzar resultados de última generación en diversos benchmarks. La clave está en que el LLM aprende de sí mismo, guiándose de sus propias respuestas, pero solo de las que genera con una 'alta confianza'.

Self-Refine [Madaan et al., 2023]: Es una técnica que permite a los grandes modelos de lenguaje (LLMs) mejorar sus propias respuestas mediante un proceso iterativo de retroalimentación y refinamiento, inspirado en cómo los humanos revisan y mejoran sus escritos. La idea central es usar un mismo LLM para generar una respuesta inicial, luego proporcionar retroalimentación sobre esa respuesta, y usar esta retroalimentación para refinarla iterativamente. Este proceso no requiere datos de entrenamiento supervisados, entrenamiento adicional o aprendizaje por refuerzo, sino que utiliza el mismo LLM como generador, refinador y proveedor de retroalimentación. De esta manera, el propio modelo se encarga de mejorar su salida usando retroalimentación de su propia salida, en un proceso iterativo de auto-mejora.

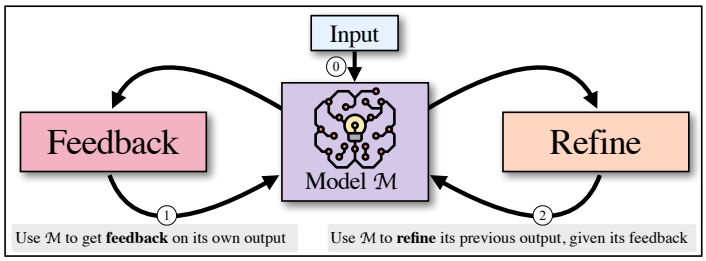

Figura 7: *Self-Refine, de [Madaan et al., 2023]*

Progressive Hint [Zheng et al., 2024]: Este trabajo propone simular la capacidad humana de auto-verificación en modelos de lenguaje (LLMs) mediante 'Progressive-Hint Prompting' (PHP). En este método, un LLM genera una respuesta inicial y luego la combina con la pregunta original para una nueva ronda de razonamiento, utilizando su propia respuesta como pista. Si la nueva respuesta coincide con la anterior, se aumenta la confianza en su corrección. El método usa prompts progresivos, que añaden una frase indicando la proximidad de la respuesta a las posibles respuestas anteriores, junto a un recordatorio de dichas respuestas. Este diseño contempla dos situaciones: (1) que las pistas sean iguales a la respuesta correcta, para asegurar que el modelo no se desvíe de la respuesta y (2) que las pistas no sean iguales a la respuesta correcta, para verificar que el modelo puede corregir una respuesta errónea.

Self-Taught Reasoner(STaR) [Zelikman et al., 2022]: Es una técnica que permite a los modelos de lenguaje (LMs) mejorar su capacidad de razonamiento mediante un pro-

ceso iterativo de auto-aprendizaje. A diferencia de los enfoques que requieren grandes conjuntos de datos con razonamientos explícitos o que sacrifican precisión con pocos ejemplos (few-shot), STaR se basa en un bucle simple. Primero, el modelo genera razonamientos (tipo 'chain-of-thought') para responder preguntas, utilizando unos pocos ejemplos. Si las respuestas son incorrectas, el modelo intenta generar el razonamiento de nuevo, pero esta vez usando la respuesta correcta como guía. Finalmente, se ajusta el modelo con todos los razonamientos que llevaron a la respuesta correcta. Este proceso se repite varias veces, permitiendo que el modelo aprenda a razonar de manera más efectiva a partir de sus propios intentos, usando un mecanismo de auto-aprendizaje para obtener una mejor capacidad de razonamiento (notar que es similar a [Huang et al., 2022], pero usando 'hints' como en [Zheng et al., 2024]).

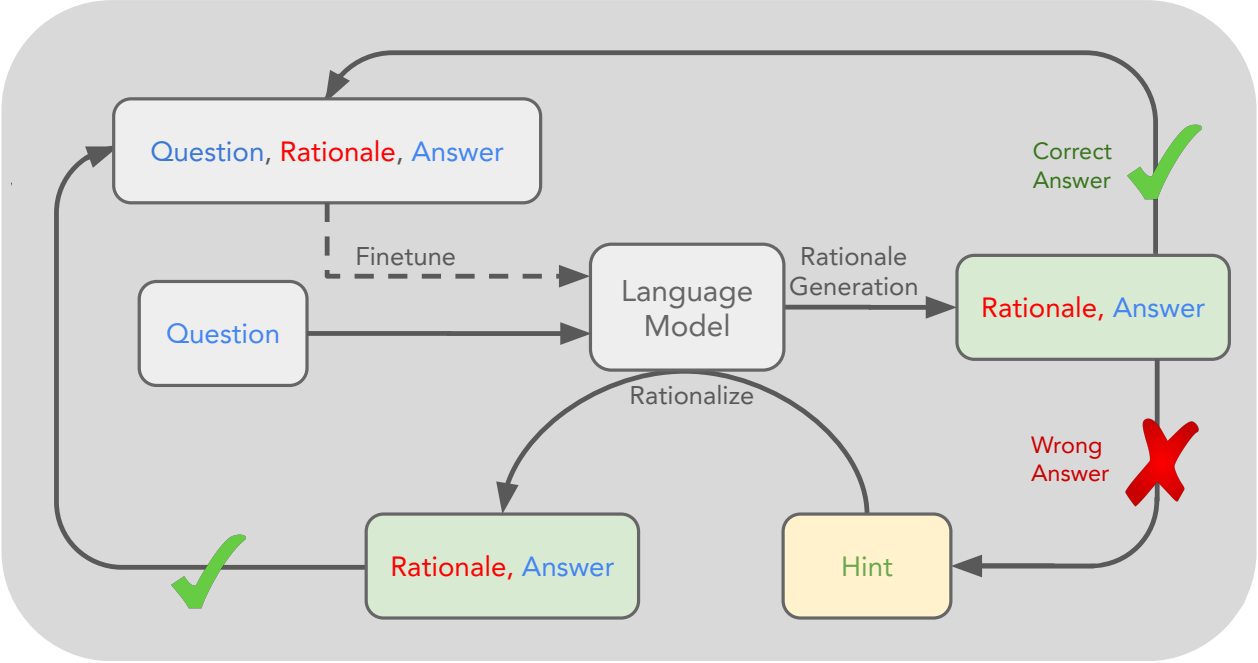

Figura 8: *STaR, de [Zelikman et al., 2022]*

Least-to-most(L2M) [Zhou et al., 2023]: Es una estrategia de prompting que busca solucionar la dificultad del 'chain-of-thought' (CoT) para generalizar a problemas más complejos que los ejemplos usados en el prompt. La idea central es descomponer un problema complejo en una secuencia de subproblemas más simples y resolverlos en orden creciente de dificultad. La solución de cada subproblema se facilita mediante las respuestas obtenidas en los subproblemas previamente resueltos.

Tree-of-Thoughts(ToT) [Yao et al., 2023]: Es un nuevo marco para la inferencia en modelos de lenguaje (LMs) que busca superar las limitaciones de los procesos de decisión a nivel de token de izquierda a derecha, que dificultan la resolución de problemas que requieren exploración, visión estratégica o decisiones iniciales clave. ToT generaliza el enfoque popular 'Chain of Thought' (CoT) y permite la exploración de múltiples 'pensamientos' (coherent units of text) que actúan como pasos intermedios hacia la solución de un problema. El marco ToT permite a los LMs tomar decisiones deliberadas al considerar diferentes caminos de razonamiento, auto-evaluar sus elecciones y decidir el siguiente paso, permitiendo una búsqueda más estratégica, e incluso retroceder cuando sea necesario para tomar decisiones globales.

Graph of Thoughts (GoT) [Besta et al., 2024]: modela el proceso de razonamiento de un LLM como un grafo arbitrario, donde los nodos representan 'pensamientos' (unidades de información generadas por el LLM) y las aristas indican dependencias entre estos pensamientos. A diferencia de 'Chain-of-Thought' (CoT), que utiliza una cadena lineal de pensamientos, o 'Tree-of-Thoughts' (ToT), que organiza los pensamientos en un árbol, GoT permite cualquier tipo de conexión entre nodos, habilitando transformaciones como la agregación (combinar múltiples pensamientos en uno), el refinamiento (iterar sobre un pensamiento para mejorarlo) y la generación de nuevos pensamientos a partir de otros existentes, sin restricciones estructurales. Esta flexibilidad permite emular procesos de razonamiento más complejos y cercanos a la cognición humana, donde las ideas se interconectan y refinan de forma no lineal.

Buffer of Thoughts(BoT) [Yang et al., 2024]: Es un nuevo enfoque de razonamiento aumentado con 'pensamientos' (thought-augmented reasoning) que busca mejorar la precisión, eficiencia y robustez de los grandes modelos de lenguaje (LLMs). BoT utiliza un 'meta-buffer' para almacenar una serie de 'plantillas de pensamiento' (thought-templates) informativas y de alto nivel, que se han extraído de los procesos de resolución de problemas en diversas tareas. Para cada nuevo problema, BoT recupera una plantilla de pensamiento relevante y la adapta con estructuras de razonamiento específicas para llevar a cabo un razonamiento eficiente. Para garantizar la escalabilidad y estabilidad, BoT incorpora un 'buffer-manager' que actualiza dinámicamente el meta-buffer a medida que se resuelven más tareas, aumentando su capacidad.

Visualization-of-Thought (VoT) [Wu et al., 2024]: es una técnica de 'prompting' inspirada en la capacidad humana de crear imágenes mentales ('ojo de la mente') para

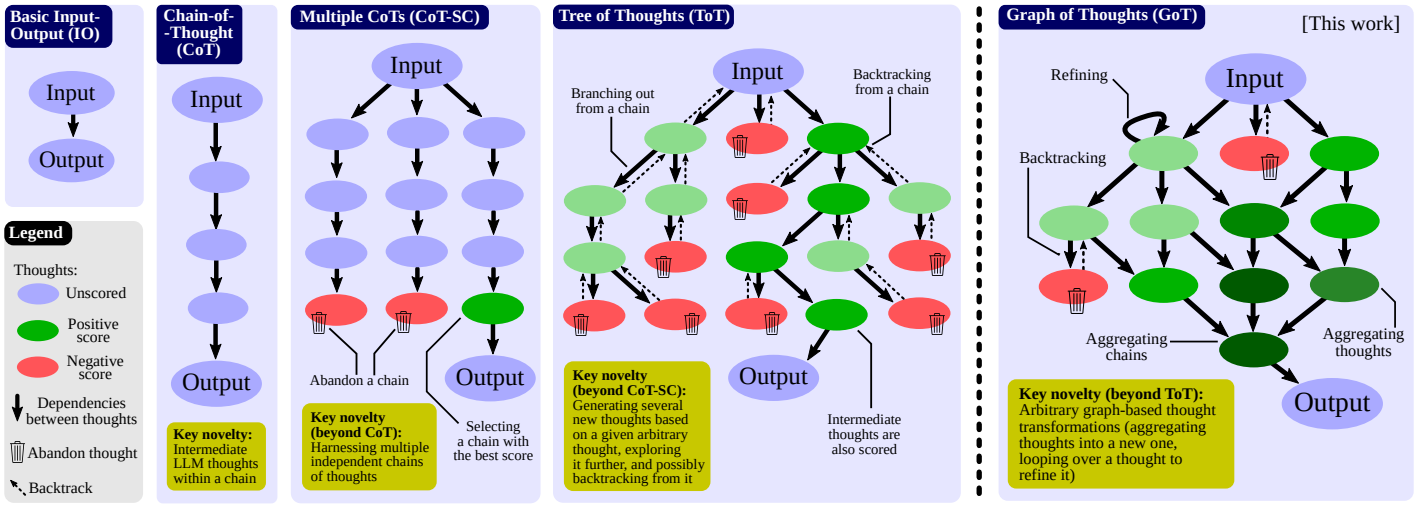

Figura 9: *Comparativa entre IO, CoT, CoT-SC, ToT y GoT, tomado de [Besta et al., 2024]*

mejorar el razonamiento espacial en grandes modelos de lenguaje (LLMs). VoT busca inducir a los LLMs a 'visualizar' sus trazas de razonamiento, guiando así los pasos de razonamiento subsiguientes. A diferencia de otros métodos que se centran en el razonamiento puramente lingüístico o simbólico, VoT introduce un componente visual explícito en el proceso. Los autores aplican VoT a tareas de razonamiento espacial multi-salto en entornos de cuadrícula 2D, incluyendo navegación en lenguaje natural, navegación visual y 'tiling' visual, demostrando mejoras significativas sobre el rendimiento de los LLMs y superando incluso a modelos de lenguaje multimodal (MLLMs) existentes. Aunque VoT se probó principalmente en LLMs, su fundamento en la visualización sugiere un potencial considerable para su aplicación en MLLMs.

Chain of Continuous Thought(COCONUT) [Hao et al., 2024]: Es un nuevo paradigma de razonamiento para grandes modelos de lenguaje (LLMs) que explora el potencial de razonar en un espacio latente sin restricciones, en lugar del espacio del lenguaje natural donde se aplica típicamente 'chain-of-thought' (CoT). En Coconut, la representación del estado del razonamiento (el 'pensamiento continuo') se obtiene del último estado oculto del LLM, pero en vez de decodificarlo a tokens de palabras, se alimenta de nuevo al LLM como la entrada subsiguiente directamente en el espacio continuo. Esto evita que los modelos se vean limitados por la necesidad de que cada paso de razonamiento sea coherente en lenguaje natural, y permite que el modelo gestione la coherencia y el razonamiento de forma separada, pudiendo incluso generar representaciones que no tienen por qué existir en el lenguaje natural.

Esta técnica lleva a patrones emergentes de razonamiento avanzado. El 'pensamiento continuo' puede codificar múltiples alternativas de pasos de razonamiento siguientes, permitiendo al modelo llevar a cabo una búsqueda en anchura (BFS) para resolver el problema, en lugar de comprometerse prematuramente con una sola ruta determinista, como hace CoT (aquí se sigue la premisa de que el lenguaje no es necesario para el razonamiento, sobre lo cual se puede leer más al respecto en [Fedorenko et al., 2024]).

Many-shot In-Context Learning [Agarwal et al., 2024]: Los Modelos de Lenguaje demuestran gran habilidad en el aprendizaje en contexto con pocos ejemplos (ICL), pero las recientes expansiones en las ventanas de contexto permiten explorar el ICL con muchos ejemplos, revelando ganancias de rendimiento significativas en diversas tareas. Dado que la disponibilidad de ejemplos humanos limita el potencial del ICL con muchos ejemplos, se investigan enfoques alternativos como el ICL Reforzado, que emplea razonamientos generados por el modelo, y el ICL No Supervisado, que solo utiliza preguntas específicas del dominio; ambos resultan efectivos, especialmente en razonamiento complejo. Además, se observa que el ICL con muchos ejemplos supera sesgos de preentrenamiento, aprende funciones complejas y compite con el fine-tuning, aunque el costo de inferencia aumenta linealmente y el beneficio varía entre LLMs avanzados, señalando las limitaciones de la pérdida de predicción del siguiente token como métrica de rendimiento en ICL.

DeepSeek-R1: un regalo a la comunidad [DeepSeek-AI et al., 2025]: El desarrollo de *DeepSeek-R1*, un modelo de lenguaje (LLM) de código abierto con notables capacidades de razonamiento, ilustra el impacto de diferentes estrategias de entrenamiento, en particular el aprendizaje por refuerzo (RL) y el ajuste fino supervisado (SFT). El proceso de desarrollo involucró tres variantes distintas:

- **DeepSeek-R1-Zero:** Este modelo se entrenó directamente con RL, utilizando el algoritmo de optimización de política relativa grupal (GRPO) [Shao et al., 2024] y recompensas basadas en la precisión y el uso de etiquetas de pensamiento, sin una fase previa de SFT. Demostró la capacidad de RL para generar capacidades de razonamiento de forma autónoma, aunque a costa de un mayor tiempo de cómputo en inferencia.
- **DeepSeek-R1:** Este modelo partió de un punto de control pre-entrenado con SFT usando datos de alta calidad diseñados por humanos. Luego, se aplicó RL de forma similar a DeepSeek-R1-Zero, mejorando aún más sus capacidades. Se incorporaron rondas adicionales de SFT con datos generados por muestreo de rechazo para aumentar la calidad y variedad. Finalmente, se realizó un ciclo de RL enfocado en la utilidad y la seguridad. Este modelo alcanzó un rendimiento comparable a modelos significativamente más grandes.
- **Distill' Models:** Estos modelos, más pequeños (basados en Qwen y Llama), se entrenaron exclusi-

vamente con SFT, utilizando los datos de razonamiento generados por DeepSeek-R1. Este proceso de *destilación* transfiere el conocimiento del modelo más grande a los modelos más pequeños, mejorando su rendimiento sin necesidad de RL.

La experiencia con DeepSeek-R1 confirma que, si bien RL puede inducir capacidades de razonamiento superiores y generalización (como se discute en [Chu et al., 2025]), SFT juega un papel crucial en la estabilización del entrenamiento y en la mejora de la calidad y eficiencia, especialmente cuando se utilizan datos de alta calidad o se aplica un proceso de destilación desde un modelo más potente. La combinación estratégica de ambas técnicas, junto con una cuidadosa selección y generación de datos, resulta fundamental para el desarrollo de LLMs con capacidades de razonamiento avanzadas.

Datasets

> *"When a measure becomes a target, it ceases to be a good measure"*
>
> *Goodhart's maxim*

Abstraction and Reasoning Corpus (ARC) [Chollet, 2019]: se presenta como un benchmark para evaluar la inteligencia artificial general, así como un test psicométrico para humanos. Inspirado en los tests de Matrices Progresivas de Raven, ARC presenta tareas abstractas en forma de cuadrículas con símbolos, donde el objetivo es inferir la transformación que mapea una serie de inputs a sus correspondientes outputs, y aplicar dicha transformación a un nuevo input. A diferencia de los tests de CI convencionales, ARC se centra en la inteligencia fluida general, evaluando la eficiencia en la adquisición de nuevas habilidades, y no la habilidad en sí misma. Para ello, mide la generalización 'consciente del desarrollador', presentando tareas únicas y desconocidas, limitando los ejemplos a unos pocos, y controlando estrictamente los priors de conocimiento, asumiendo solo los priors innatos de Core Knowledge y ningún otro conocimiento adquirido.

Algunas preguntas, no muy sencillas: El documento *Easy Problems That LLMs Get Wrong* [Williams and Huckle, 2024] presenta un *benchmark* lingüístico diseñado para evaluar las *limitaciones* de los grandes modelos de lenguaje (LLMs) en áreas donde los humanos, en general, muestran facilidad, como el razonamiento lógico, espacial, relacional, la comprensión lingüística, conocimiento científico popular y el sentido común. A través de preguntas sencillas, el *benchmark* revela que los LLMs, a pesar de su entrenamiento masivo, a menudo fallan debido a la sobreespecialización en los datos de entrenamiento (*overfitting*) y dificultades con la generalización.

Un claro ejemplo de esta sobreespecialización se encuentra en una variación del problema de Monty Hall. La pregunta, junto con una respuesta incorrecta típica de un LLM (en este caso, Claude 3 Opus) se aprecia en:

> **Pregunta:** Suppose you're on a game show, and you're given the choice of three doors: Behind one door is a gold bar; behind the others, rotten vegetables. You pick a door, say No. 1, and the host asks you "Do you want to pick door No. 2 instead?" Is it to your advantage to switch your choice?
>
> **Respuesta de Claude 3 Opus:** "This is a classic probability problem known as the "Monty Hall problem." The answer is that it is always to your advantage to switch your choice. Here's why: ..."

Como se observa arriba, el LLM responde incorrectamente, aplicando el razonamiento estándar del problema de Monty Hall clásico, a pesar de que la pregunta modificada 'elimina' la acción clave del presentador (abrir una puerta mostrando que hay detrás). Esto demuestra una clara incapacidad para adaptar sus 'conocimientos previos' a una situación con condiciones diferentes (solo por probar le hicimos la misma pregunta a *Gemini Flash Thinking 2*, *o3-mini* del *free-tier*, y a *DeepSeek-R1*, de los cuales solo R1 respondió correctamente a la primera sin especificarle que no use la respuesta de Monty Hall). Esta dificultad para manejar variaciones de problemas conocidos resalta la importancia de considerar los 'priors' de conocimiento, como los descritos por Chollet en su trabajo sobre la medición de la inteligencia, a la hora de diseñar y evaluar LLMs.

Además de la sobreespecialización ejemplificada con el problema de Monty Hall, el estudio identifica otros fallos comunes en los LLMs, como la falta de lógica o sentido común en las respuestas, dificultades con el razonamiento espacial, errores en cálculos matemáticos simples y una comprensión lingüística superficial que no capta matices o restricciones específicas.

RuleTaker [Clark et al., 2020]: es un *benchmark* diseñado para evaluar el razonamiento lógico deductivo en modelos de lenguaje. Se centra en la capacidad del modelo para aplicar reglas lógicas explícitas, presentadas en lenguaje natural, a un conjunto de hechos para derivar conclusiones. A diferencia de tareas que se basan en conocimiento implícito o estadístico, RuleTaker exige una comprensión precisa de la estructura lógica y la aplicación rigurosa de reglas. La dificultad radica en que los LLMs, a menudo, tienen problemas para generalizar reglas lógicas más allá de ejemplos muy simples y para mantener la consistencia en la aplicación de dichas reglas, especialmente a medida que aumenta la complejidad de las inferencias. Un ejemplo prototípico sería: *Si todos los humanos son mortales y Juan es humano, ¿es Juan mortal?*.

EntailmentBank [Dalvi et al., 2021]: evalúa el razonamiento textual y la capacidad de deducción de los LLMs. A diferencia de la simple clasificación de implicación (entailment), EntailmentBank requiere que el modelo genere una cadena de razonamiento que conecte una serie de premisas con una conclusión. Esto implica no solo determinar si la conclusión se sigue de las premisas, sino también explicar el razonamiento paso a paso. La dificultad para los LLMs reside en la necesidad de seguir correctamente las inferencias intermedias y de construir una cadena de razonamiento coherente y completa. Un ejemplo ilustrativo sería:

- Premisa 1: Todos los pájaros vuelan.
- Premisa 2: Un pingüino es un pájaro.
- Conclusión: ¿Vuela el pingüino?.

GSM8K (Grade School Math 8K) [Cobbe et al., 2021]: es un *benchmark* que contiene problemas matemáticos verbales diseñados para estudiantes de primaria. Aunque los problemas son conceptualmente sencillos para los humanos, representan un desafío significativo para los LLMs porque requieren una combinación de comprensión del lenguaje natural, razonamiento paso a paso y cálculos aritméticos precisos. Los LLMs a menudo fallan debido a errores en la interpretación del problema, dificultades para planificar la secuencia correcta de operaciones o errores en los cálculos, incluso en operaciones básicas. Un ejemplo típico sería: *Si Juan tiene 3 manzanas y compra 5 más, pero luego regala la mitad, ¿cuántas le quedan?*.

MathQA [Amini et al., 2019]: es un *benchmark* que evalúa el razonamiento matemático en un formato de opción múltiple. Abarca una gama más amplia de problemas matemáticos que GSM8K, incluyendo álgebra, geometría y otros temas, y presenta un mayor nivel de complejidad. La dificultad para los LLMs radica en la necesidad de integrar la comprensión del lenguaje natural (para interpretar el problema verbal) con habilidades matemáticas más avanzadas. Los errores suelen surgir de una interpretación incorrecta del problema, errores en la aplicación de fórmulas o fallos en los cálculos. Un ejemplo sería: *¿Cuál es el área de un triángulo con base 10 y altura 5? (A) 25, (B) 50, (C) 75.*.

DROP (Discrete Reasoning Over Paragraphs) [Dua et al., 2019]: es un *benchmark* de comprensión lectora que pone un énfasis especial en el razonamiento textual discreto. A diferencia de las tareas de comprensión lectora tradicionales, DROP requiere que los LLMs identifiquen y combinen información dispersa a lo largo de un párrafo, a menudo realizando operaciones aritméticas o lógicas basadas en esa información. La dificultad para los LLMs radica tanto en la comprensión del texto como en la capacidad de realizar razonamientos discretos sobre la información extraída. Un ejemplo característico es: *¿Cuántos años pasaron entre el evento A y el evento B?*, donde la respuesta requiere extraer fechas de diferentes partes del texto y calcular la diferencia.

CommonsenseQA [Talmor et al., 2019]: evalúa el razonamiento basado en el sentido común en LLMs. Presenta preguntas de opción múltiple que, para los humanos, son triviales de responder gracias a nuestro conocimiento implícito del mundo. Sin embargo, para los LLMs, que carecen de esta experiencia encarnada y conocimiento del mundo real, estas preguntas representan un desafío significativo. Los errores suelen deberse a sesgos en los datos de entrenamiento o a la incapacidad del modelo para inferir relaciones implícitas entre conceptos. Un ejemplo sería: *¿Qué usas para cortar un pastel? (A) Tijeras, (B) Cuchillo, (C) Martillo.*.

GPQA (Graduate-Level Google-Proof Q&A) [Rein et al., 2023]: es un *benchmark* diseñado para evaluar la capacidad de los grandes modelos de lenguaje (LLMs) para responder preguntas extremadamente desafiantes, a nivel de posgrado, en los dominios de biología, física y química. Estas preguntas están diseñadas para ser *a prueba de Google*, lo que significa que una simple búsqueda en internet no es suficiente para encontrar la respuesta. Requieren una comprensión profunda y especializada del tema, así como la capacidad de razonar sobre conceptos complejos. La dificultad es tal que incluso expertos humanos en estos campos alcanzan una precisión de alrededor del 65 %. GPQA es útil para experimentos de supervisión escalable, donde los sistemas de IA necesitan proporcionar información confiable que vaya más allá de las capacidades humanas en dominios específicos.

MMLU (Massive Multitask Language Understanding) [Hendrycks et al., 2021a]: MMLU es un *benchmark* amplio que evalúa tanto el conocimiento general como las habilidades de resolución de problemas de los LLMs en 57 temas diferentes, que abarcan desde matemáticas elementales hasta campos profesionales como el derecho y la ética. A diferencia de los benchmarks que se centran en un solo dominio, MMLU proporciona una evaluación exhaustiva de la comprensión del lenguaje en un contexto multitarea. Se evalúa a los modelos en escenarios *zero-shot* y *few-shot*, simulando situaciones del mundo real donde los modelos deben operar con un contexto mínimo o con solo unos pocos ejemplos. La evaluación se basa en la precisión en la respuesta a preguntas de opción múltiple.

MMLU-Pro [Wang et al., 2024]: es una versión mejorada y más desafiante de MMLU. Introduce preguntas que requieren un razonamiento más profundo y aumenta el número de opciones de respuesta de cuatro a diez. Este incremento en la complejidad reduce significativamente la probabilidad de obtener respuestas correctas por conjetura aleatoria. Además, MMLU-Pro demuestra una mayor estabilidad bajo diferentes indicaciones (prompts), lo que significa que es menos sensible a pequeñas variaciones en la formulación de las preguntas. Como consecuencia de su mayor dificultad, se observa una caída significativa en la precisión de los modelos en comparación con el MMLU original.

MATH [Hendrycks et al., 2021b]: se centra específicamente en la evaluación de la capacidad de los LLMs para resolver problemas matemáticos complejos. Abarca problemas que van desde el nivel de secundaria hasta el nivel de competición, incluyendo áreas como álgebra, geometría, probabilidad y cálculo. Cada problema en MATH viene acompañado de una solución detallada paso a paso, lo que permite no solo evaluar la respuesta final, sino también el proceso de razonamiento del modelo. MATH es particularmente relevante para aplicaciones educativas y otras áreas donde la resolución precisa y eficiente de problemas matemáticos es crucial.

HumanEval [Chen et al., 2021]: HumanEval evalúa la capacidad de los LLMs para generar código funcionalmente correcto. A diferencia de los benchmarks anteriores que se centran en el razonamiento en lenguaje natural, HumanEval presenta desafíos de programación en forma de *docstrings* (descripciones de la función en lenguaje natural) y evalúa si el código generado por el modelo pasa un conjunto de pruebas unitarias. Utiliza la métrica *pass@k*, donde se generan k soluciones de código diferentes, y el modelo se considera exitoso si al menos una de ellas pasa todas las pruebas.

Referencias

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Ode-

na. Show your work: Scratchpads for intermediate computation with language models, 2021. URL https://arxiv.org/abs/2112.00114.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023. URL https://arxiv.org/abs/2205.11916.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning, 2023. URL https://arxiv.org/abs/2210.00720.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models, 2023. URL https://arxiv.org/abs/2210.03350.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. Large language models are better reasoners with self-verification, 2023. URL https://arxiv.org/abs/2212.09561.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL https://arxiv.org/abs/2203.11171.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug, 2023a. URL https://arxiv.org/abs/2304.05128.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks, 2023b. URL https://arxiv.org/abs/2211.12588.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models, 2023. URL https://arxiv.org/abs/2211.10435.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. Refiner: Reasoning feedback on intermediate representations, 2024. URL https://arxiv.org/abs/2304.01904.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve, 2022. URL https://arxiv.org/abs/2210.11610.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023. URL https://arxiv.org/abs/2303.17651.

Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. Progressive-hint prompting improves reasoning in large language models, 2024. URL https://arxiv.org/abs/2304.09797.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning, 2022. URL https://arxiv.org/abs/2203.14465.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models, 2023. URL https://arxiv.org/abs/2205.10625.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL https://arxiv.org/abs/2305.10601.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690, March 2024. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v38i16.29720.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E. Gonzalez, and Bin Cui. Buffer of thoughts: Thought-augmented reasoning with large language models, 2024. URL https://arxiv.org/abs/2406.04271.

Wenshan Wu, Shaoguang Mao, Yadong Zhang, Yan Xia, Li Dong, Lei Cui, and Furu Wei. Mind's Eye of LLMs: Visualization-of-Thought Elicits Spatial Reasoning in Large Language Models. 2024.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2024. URL https://arxiv.org/abs/2412.06769.

Evelina Fedorenko, Steven T. Piantadosi, and Edward A. F. Gibson. Language is primarily a tool for communication rather than thought, 2024. URL https://doi.org/10.1038/s41586-024-07522-w.

Rishabh Agarwal, Avi Singh, Lei M. Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, John D. Co-Reyes, Eric Chu, Feryal Behbahani, Aleksandra Faust, and Hugo Larochelle. Many-shot in-context learning, 2024. URL https://arxiv.org/abs/2404.11018.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang,

Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training, 2025. URL https://arxiv.org/abs/2501.17161.

François Chollet. On the measure of intelligence, 2019. URL https://arxiv.org/abs/1911.01547.

Sean Williams and James Huckle. Easy Problems That LLMs Get Wrong, May 2024.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. Transformers as soft reasoners over language. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3882–3890. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/537. URL https://doi.org/10.24963/ijcai.2020/537. Main track.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. Explaining answers with entailment trees. *EMNLP*, 2021.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms, 2019. URL https://arxiv.org/abs/1905.13319.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs, 2019. URL https://arxiv.org/abs/1903.00161.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof qa benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021a. URL https://arxiv.org/abs/2009.03300.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024. URL https://arxiv.org/abs/2406.01574.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021b. URL https://arxiv.org/abs/2103.03874.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.