

RAZONAMIENTO EN GRANDES MODELOS DE LENGUAJE

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

Resumen: Este trabajo presenta una exploración del estado del arte en razonamiento con Grandes Modelos de Lenguaje (LLMs), destacando resultados clave y las tendencias más relevantes en la materia. Adicionalmente, se llevan a cabo experimentos prácticos utilizando LLMs, con un enfoque en las técnicas de ICL (In-Context Learning) y Prompt Engineering. Finalmente, se propone una estrategia para mejorar la consistencia en el razonamiento de LLMs, basada en el uso de Aprendizaje por Refuerzo.

Introducción

Este trabajo aborda la exploración y mejora del razonamiento en grandes modelos de lenguaje (LLMs) mediante un enfoque estructurado en tres etapas. En primer lugar, se presenta una revisión del estado del arte con un objetivo didáctico, buscando exponer a un alto nivel de abstracción las técnicas más comúnmente utilizadas para optimizar el desempeño de los LLMs en tareas de razonamiento.

Posteriormente, se procede a la experimentación práctica con la API de Gemini de Google. En esta sección, se ponen a prueba las ideas y conceptos extraídos de

la revisión del estado del arte, aplicando las técnicas de aprendizaje en contexto y diseño de prompts a la resolución de tareas de razonamiento con Gemini.

Finalmente, se propone una técnica basada en Aprendizaje por Refuerzo (RL) que emplea el seguimiento del código para mejorar la consistencia de las cadenas de razonamiento generadas por los LLMs. Esta estrategia busca aumentar la fiabilidad y la coherencia de los resultados, permitiendo que los LLMs proporcionen soluciones más robustas y confiables.

Revisión del state-of-the-art

A continuación, técnicas relevantes que rescatamos:

scratchpad[1]: 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.

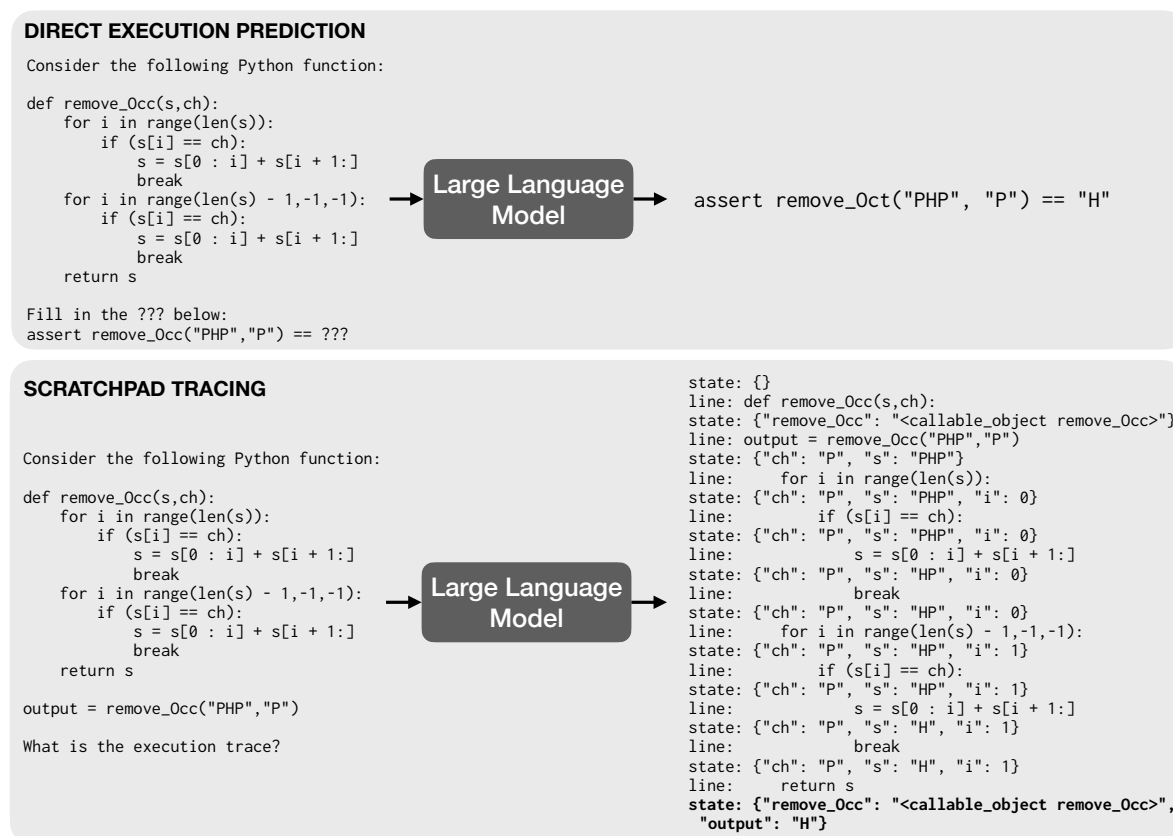


Figura 1: Ejemplo de aplicación de scratchpad a tarea de predicción de ejecución de código.

Chain of Thought(CoT)[2]: 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.

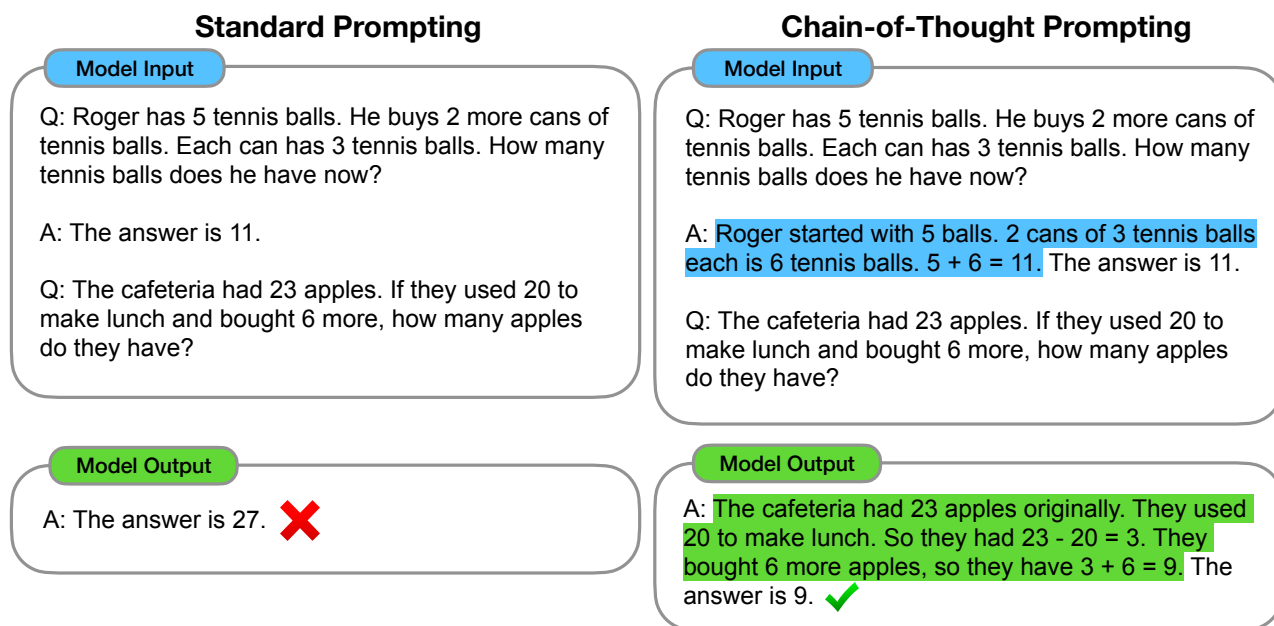


Figura 2: CoT en tarea de razonamiento matemático.

Zero-Shot CoT[3]: 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 explicita los pasos intermedios del razonamiento, lo cual mejora drásticamente su rendimiento en tareas complejas.

Complexity-Based Prompting[4]: 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[5]: 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.

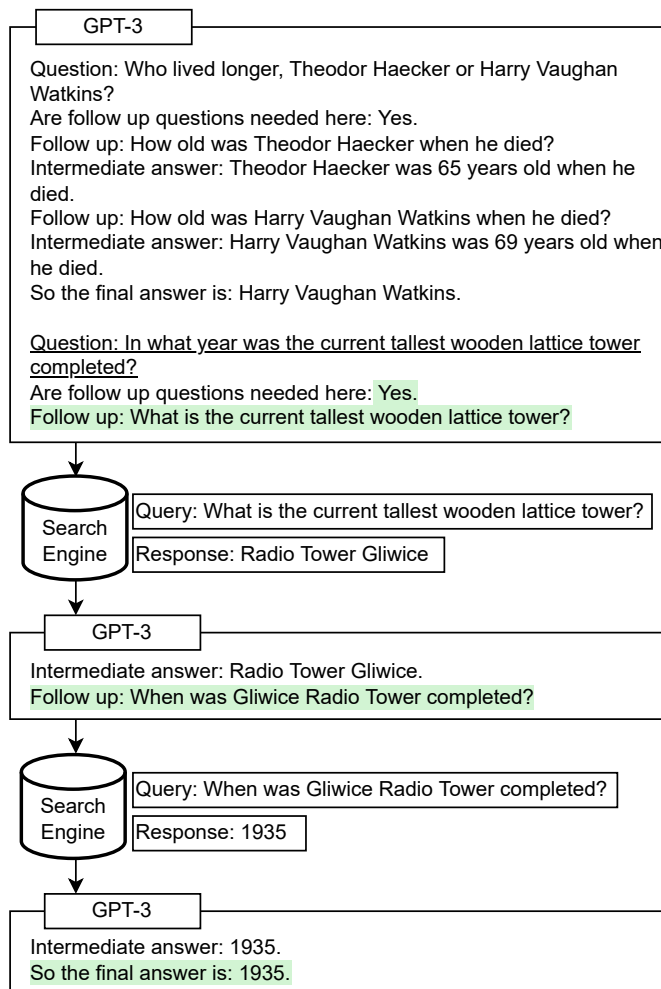


Figura 3: Flujo de Self-Ask.

Self-Verification[6]: 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.

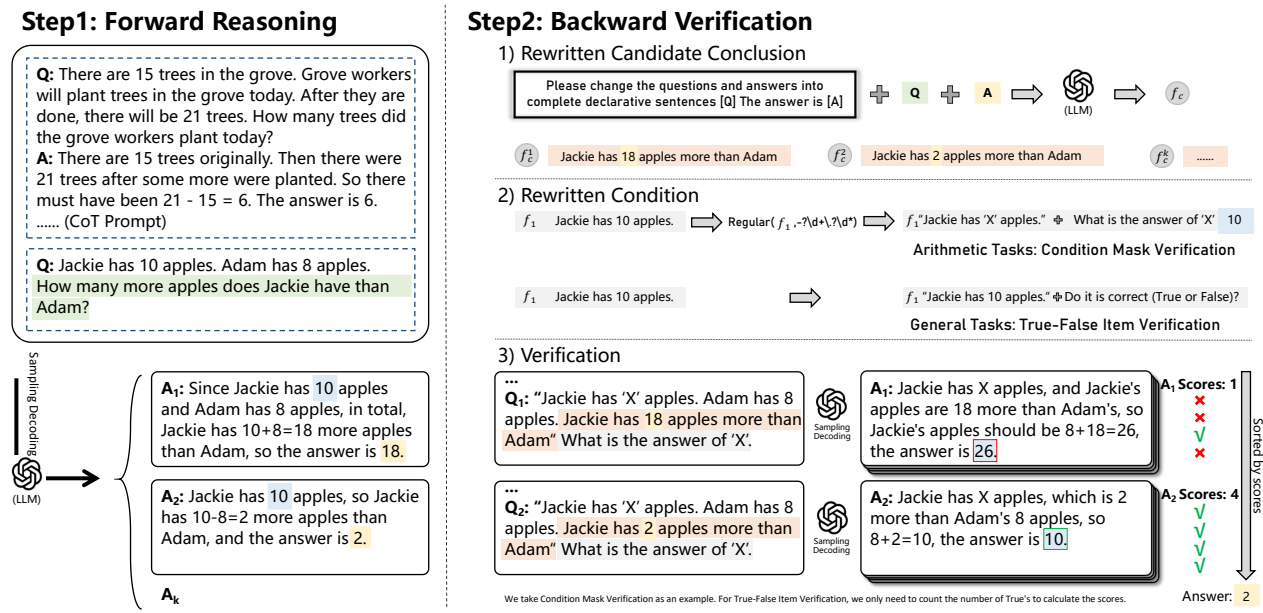


Figure 4: Self-Verification.

Self-Consistency[7]: 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[8]: 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)[9]: 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)[10]).

REFINER[11]: 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.

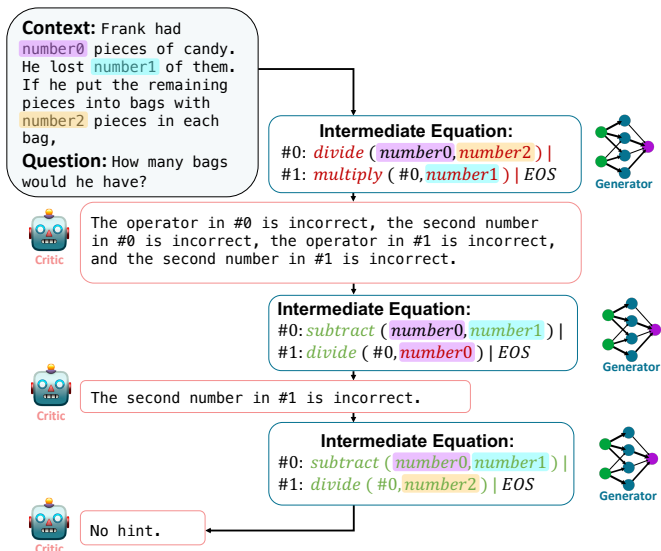
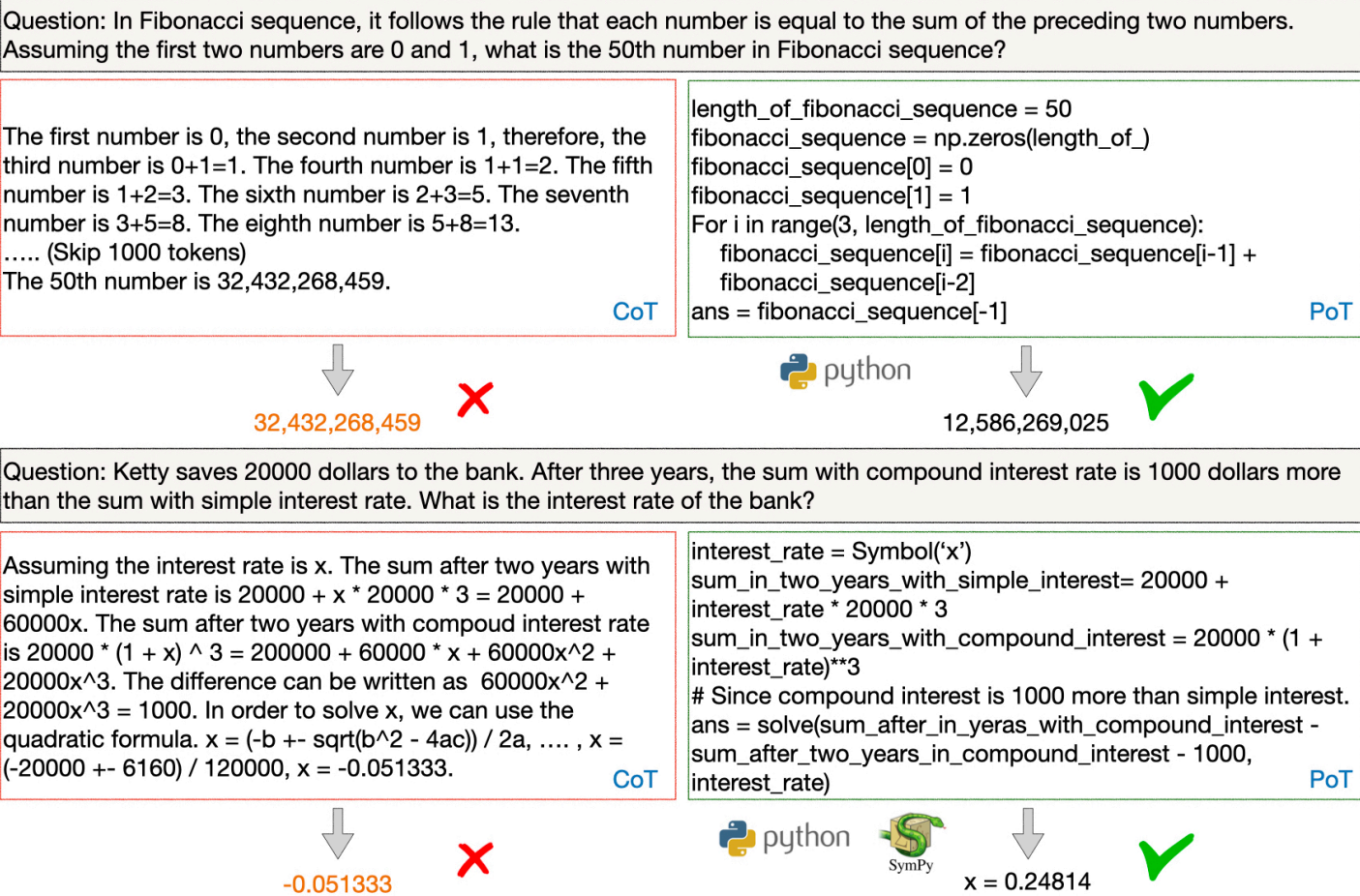


Figure 5: REFINER

Ver también 'Self-Correct' en [12] para un acercamiento similar.



Self-Taught Reasoner(STaR)[16] (ojito con este): Es una técnica que permite a los modelos de lenguaje (LMs) mejorar su capacidad de razonamiento mediante un proceso 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 in-

correctas, 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 [13], pero usando 'hints' como en [15]).

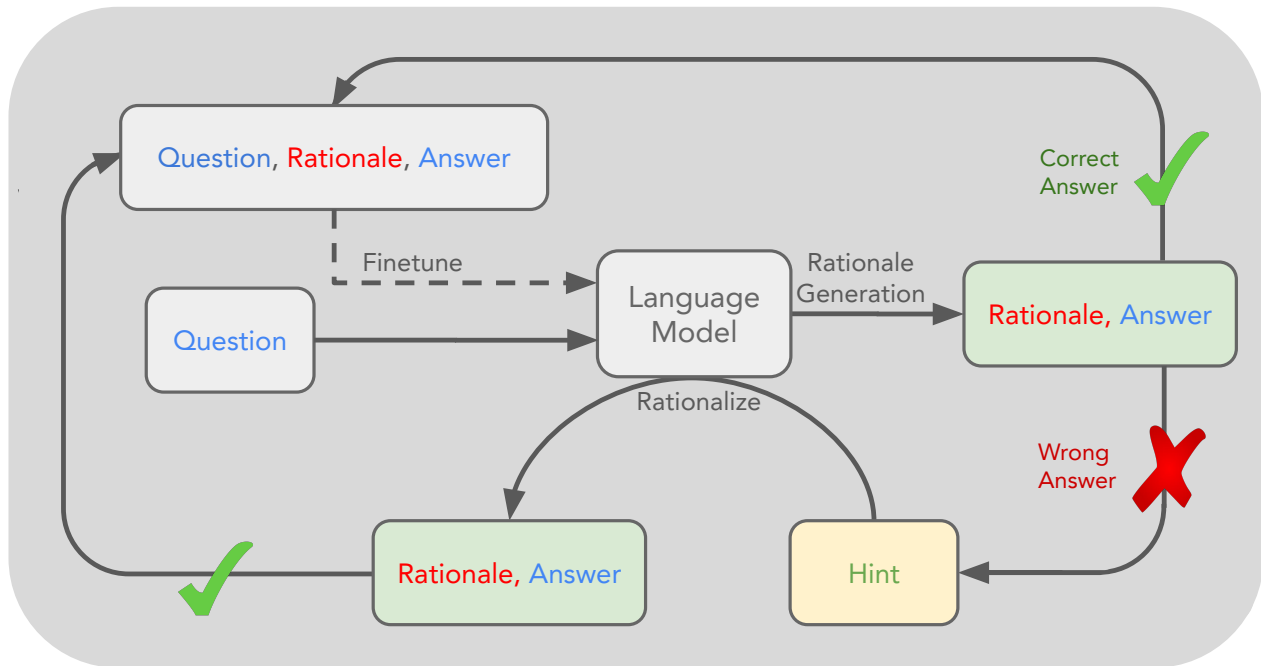


Figura 8: STaR

Least-to-most(L2M)[17]: 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)[18]: 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.

Buffer of Thoughts(BoT)[19]: 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.

Chain of Continuous Thought(COCONUT)[20]: 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 co-

dificar 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

solamente 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 [21]).

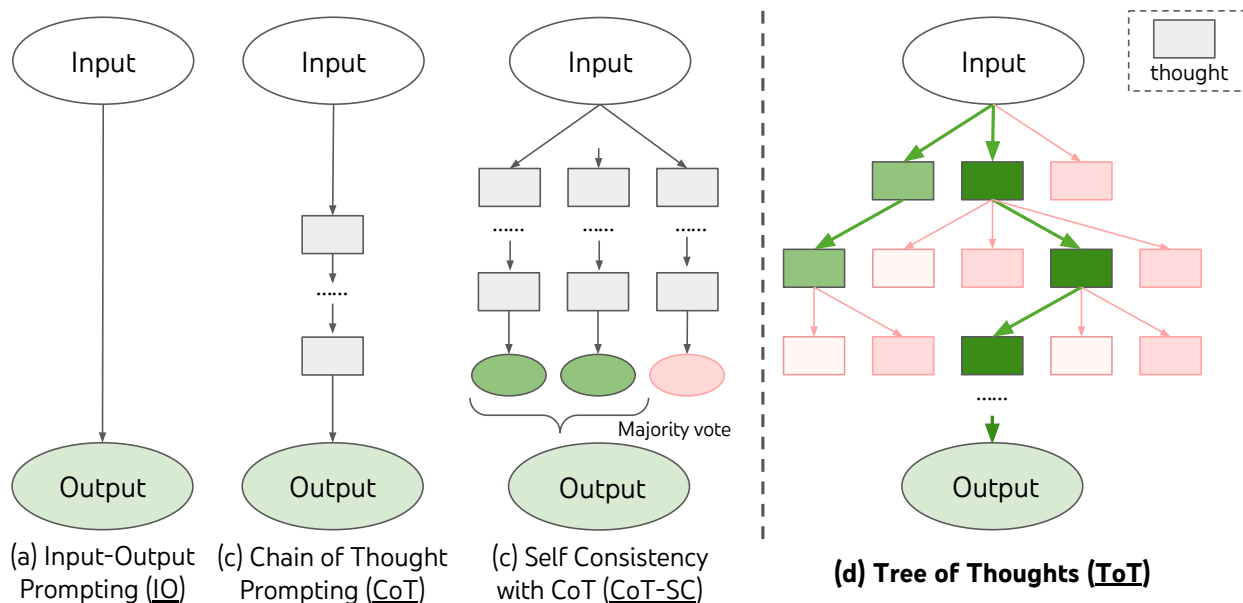


Figura 9: ToT

Nuestro humilde dataset

Para evaluar con precisión la capacidad de razonamiento de los modelos, hemos creado un dataset que incluye preguntas de tres categorías clave: lógica, combinatoria, teoría de números y predicción de salida de código. En las preguntas de combinatoria, hemos utilizado problemas que buscan derivar una fórmula cerrada, que luego debe ser evaluada en un valor numérico específico. Las preguntas de predicción de salida de código consisten en funciones con operaciones aleatorias, algunas de las cuales pueden o no cumplir una función real, lo que puede dificultar la comprensión del código por parte del modelo. En las preguntas de lógica, hemos incluido problemas del estilo de truhanes y caballeros, así como otros problemas similares. En el caso de las preguntas de lógica, se ha utilizado el formato de selección múltiple, lo que permite una evaluación objetiva de la respuesta correcta. En las categorías de combinatoria, teoría de números y predicción de salida de código, se ha especificado que la respuesta debe ser un entero, facilitando una evaluación precisa del rendimiento del modelo en estos dominios. La estructura del dataset garantiza que la respuesta de cada pregunta sea clara, específica y cuantificable, evitando la ambigüedad y permitiendo una evaluación objetiva del rendimiento.

SFT vs RL: una gran aventura, pero a que costo (?)

Como se analiza en [22], RL, con recompensas basadas en resultados, generaliza mejor que SFT en dominios textuales y visuales, mientras que SFT tiende a memorizar los datos de entrenamiento. Esta diferencia se basa en la intuición de que RL guía al modelo a buscar la forma de plantearse la problemática para alcanzar la solución, mientras que SFT se enfoca en imitar los ejemplos dados. Sin embargo, SFT sigue siendo esencial

para estabilizar el entrenamiento de RL y permitir que alcance su potencial de generalización.

Para entender mejor podemos citar el proceso de desarrollo del modelo de razonamiento DeepSeek-R1[23], un LLM que destaca por sus capacidades en diversas tareas complejas y que es de código abierto.

DeepSeek-R1-Zero: se crea aplicando directamente el aprendizaje por refuerzo (RL) a un modelo base (DeepSeek-V3-Base), sin un paso previo de ajuste fino supervisado (SFT). Este modelo utiliza el algoritmo de optimización de política relativa grupal (GRPO)[24] y se entrena con recompensas basadas en reglas para la precisión de las respuestas y el uso de etiquetas <think>y </think> en su razonamiento. El resultado es un modelo que aprende a resolver tareas de razonamiento por sí mismo, utilizando un tiempo de computación extendido en tiempo de inferencia y desarrollando comportamientos complejos, lo que demuestra el potencial del RL para generar capacidades de razonamiento de manera autónoma.

DeepSeek-R1: se crea a partir de un punto de control del modelo base que ha sido ajustado con datos de 'arranque en frío' (cold-start data) de alta calidad diseñados por humanos, para mejorar su estabilidad. Luego, se entrena con RL de manera similar a DeepSeek-R1-Zero, para mejorar sus capacidades de razonamiento. Para mejorar la calidad y variedad de sus datos, se crea una nueva ronda de datos de SFT mediante un proceso de muestreo de rechazo y usando diferentes fuentes, y con esos datos se ajusta de nuevo el modelo. Finalmente, se aplica un ciclo de RL en el que se busca que el modelo sea útil y no dañino. Este modelo resultante alcanza un rendimiento comparable a OpenAI-o1-1217, lo que demuestra la efectividad de la combinación de RL con datos de alta calidad y SFT.

Distill' Models: se crean utilizando los datos de razonamiento generados por DeepSeek-R1 para ajustar modelos base más pequeños, como Qwen y Llama, mediante un proceso de ajuste fino supervisado (SFT). Este enfoque de destilación transfiere las capacidades de razonamiento de modelos más grandes a modelos más pequeños, mejorando su rendimiento de manera eficiente. Aunque el proceso de destilación se hace solo con SFT y sin usar RL, los modelos resultantes logran superar el rendimiento de modelos similares y obtener resultados comparables a otros modelos propios.

Tenemos casos donde se usa RL, RL junto a SFT, y solamente SFT. Por una parte podemos verificar que el RL es lo que le permite al modelo superarse, y el SFT, refinarse. Podemos imaginar al pobre modelo sufriendo para mejorar sin importarle las vías, llegando a un alto intelecto, pero siendo regado, por todo lo que ha probado, y luego organizando sus ideas y practicando como decirlas, imitando los casos donde mejor lo ha hecho.

Proponemos: para mejorar la consistencia en las cadenas de pensamiento de estos modelos, el uso de aprendizaje por refuerzo (RL) en la predicción de código. Tenemos la posibilidad de generar cientos de datos de códigos distintos, e intrincados, y para predecirlos, el modelo debe seguir consistentemente cada paso. Alcanzar un 'accuracy' alto implicaría que el modelo ha seguido la ejecución del programa como una Máquina de Turing, y debió de aprender la forma de pensar en la cual garantiza esta consistencia.

Trabajos pendientes para un futuro (quizás cercano)

Para futuras investigaciones, quedan por explorar diversas estrategias para la mejora del razonamiento en modelos de lenguaje. Entre ellas, se encuentra la integración de los Large Concept Models[25], así como la exploración de su posible combinación con técnicas como Buffer of Thoughts o incluso el paradigma de razonamiento en espacio latente de Coconut. Estas líneas de trabajo ofrecen un amplio abanico de posibilidades para potenciar la capacidad de los modelos en tareas complejas, combinando nuevos conceptos con los avances actuales en diferentes enfoques de razonamiento.

El enfoque de seguimiento de código, cuando se aplica a código que transforma matrices, puede ser una estrategia efectiva para mejorar la consistencia en las modificaciones realizadas por los modelos en desafíos como ARC-AGI. Al guiar al modelo a explicitar los pasos de transformación matricial, se incrementa la probabilidad de éxito en la tarea, donde el razonamiento espacial es particularmente importante. A su vez, esta estrategia puede suplementar a modelos multimodales que deben extraer correctamente los patrones de color presentes en un grid, una capacidad donde estos modelos aún muestran dificultades, y es inevitablemente de importancia el avance en el razonamiento que evalúa este benchmark.

Referencias

- [1] 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 Odena. Show your work: Scratchpads for intermediate computation with language models, 2021.
- [2] 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.
- [3] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023.
- [4] Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning, 2023.
- [5] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models, 2023.
- [6] 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.
- [7] 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.
- [8] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug, 2023.
- [9] Wenhui Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks, 2023.
- [10] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models, 2023.
- [11] Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. Refiner: Reasoning feedback on intermediate representations, 2024.
- [12] Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct, 2022.
- [13] Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can self-improve, 2022.
- [14] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, 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.
- [15] Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhen-guo Li, and Yu Li. Progressive-hint prompting improves reasoning in large language models, 2024.
- [16] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning, 2022.

- [17] 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.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] Evelina Fedorenko, Steven T. Piantadosi, and Edward A. F. Gibson. Language is primarily a tool for communication rather than thought, 2024.
- [22] 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.
- [23] 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, Fucang 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, Miaojuan 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, Wanbiao 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.
- [24] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseek-math: Pushing the limits of mathematical reasoning in open language models, 2024.
- [25] LCM team, Loïc Barrault, Paul-Ambroise Duquenne, Maha Elbayad, Artyom Kozhevnikov, Belen Alastruey, Pierre Andrews, Mariano Coria, Guillaume Couairon, Marta R. Costa-jussà, David Dale, Hady Elsahar, Kevin Heffernan, João Maria Janeiro, Tuan Tran, Christophe Ropers, Eduardo Sánchez, Robin San Roman, Alexandre Mourachko, Safiyyah Saleem, and Holger Schwenk. Large concept models: Language modeling in a sentence representation space, 2024.