Codexi: An AI-Based Adaptive Tutor for Guided Learning of Algorithms

Lucas José Paganini
Universidad Tecnológica Nacional – Facultad Regional La Plata, Bs. As., Argentina
E-mail: lucasjpaganini@gmail.com

Abstract. Learning algorithms is usually a hard step for students who are just starting with programming. While generative AI tools make it easy to get solutions, they often remove part of the thinking process, which ends up affecting how students approach problems.

This work introduces an AI-based tutor that guides learning rather than simply providing solutions. Instead, it supports students through a sequence of steps: understanding the problem, reflecting on the data involved, outlining a solution, and only then writing code. The goal is simple: to slow the process slightly so students take time to think before coding.

The system also keeps a basic record of how each student is doing. Based on that, it highlights what seems to be going well and what still needs practice, adjusting the feedback as they move forward. We also tried an interview-style interaction, where the system asks questions instead of giving hints, which changes how students engage with the task.

From early use, we noticed that students tend to organize their ideas more clearly and rely less on full answers. These are still initial observations, but they suggest that guiding the process may be more helpful than simply providing solutions.

Keywords: algorithm learning, intelligent tutoring systems, generative AI, programming education, guided learning.

Codexi: Un tutor adaptativo basado en IA para el aprendizaje guiado de algoritmos

Resumen. El aprendizaje de algoritmos suele ser una etapa difícil para quienes están empezando a programar. Si bien las herramientas de inteligencia artificial generativa facilitan obtener soluciones, muchas veces reducen parte del proceso de razonamiento, lo que termina afectando la forma en que los estudiantes abordan los problemas.

Este trabajo introduce un tutor basado en inteligencia artificial que guía el aprendizaje en lugar de limitarse a ofrecer soluciones. En su lugar, acompaña al estudiante a través de una serie de pasos: comprender el problema, reflexionar sobre los datos involucrados, plantear una solución y recién después escribir el

código. La idea es simple: desacelerar el proceso para que el estudiante piense antes de programar.

El sistema también mantiene un registro básico del desempeño de cada usuario. A partir de eso, señala qué aspectos parecen estar bien y cuáles necesitan más práctica, ajustando la retroalimentación a medida que avanza. Además, se probó un modo de interacción tipo entrevista, donde el sistema hace preguntas en lugar de dar pistas, lo que cambia la forma en que el estudiante se enfrenta al ejercicio.

En las primeras pruebas se observó que los estudiantes tienden a organizar mejor sus ideas y a depender menos de respuestas completas. Aunque son observaciones iniciales, sugieren que guiar el proceso puede ser más útil que simplemente dar soluciones.

Palabras clave: tutor inteligente, aprendizaje de algoritmos, inteligencia artificial, aprendizaje adaptativo, retroalimentación automática.

1 Introducción

Aprender algoritmos suele ser uno de los primeros obstáculos reales para quienes empiezan a programar. Más allá de la sintaxis, lo que aparece como dificultad es cómo pensar un problema: por dónde empezar, qué información es relevante y cómo ordenar una solución paso a paso.

En los últimos años surgieron herramientas de inteligencia artificial que pueden resolver este tipo de ejercicios en cuestión de segundos, lo cual cambió bastante la dinámica de aprendizaje. Muchos estudiantes ya no se quedan tanto tiempo intentando resolver un problema, porque saben que pueden obtener una respuesta completa casi de inmediato. A partir de esto, surge una pregunta bastante directa: ¿Cómo aprovechar estas herramientas sin que reemplacen el proceso de aprendizaje? En lugar de usarlas solo para obtener resultados, parece más interesante pensar en cómo pueden acompañar el proceso, ayudando a que el estudiante construya su propia solución.

En los enfoques tradicionales de enseñanza de algoritmos, el proceso de aprendizaje suele centrarse en la resolución de ejercicios a partir de explicaciones teóricas seguidas de práctica guiada por el docente. Si bien este modelo permite introducir los conceptos fundamentales, frecuentemente presenta limitaciones en el acompañamiento individual del proceso de razonamiento de cada estudiante, especialmente en etapas iniciales. En este contexto, la disponibilidad de herramientas basadas en inteligencia artificial ha facilitado el acceso a soluciones inmediatas, pero en muchos casos a costa de reducir la participación activa en la construcción del conocimiento. Frente a esto, la propuesta de este trabajo se orienta a recuperar el valor del proceso, incorporando una guía estructurada que promueve la reflexión progresiva antes de la implementación.

La Fig. 1 sintetiza la propuesta metodológica de Codexi, basada en un flujo de aprendizaje estructurado donde el Tutor de IA acompaña transversalmente cada etapa del proceso. El diseño promueve una resolución guiada y progresiva, priorizando la comprensión antes de la implementación y favoreciendo la construcción autónoma de la solución.

2 Trabajo relacionado

El uso de herramientas para acompañar el aprendizaje de programación tiene bastante recorrido. Desde hace años existen sistemas de tutoría inteligente que buscan ayudar al estudiante mientras resuelve problemas, ya sea ofreciendo pistas, señalando errores o ajustando la dificultad en función de su desempeño (Anderson et al., 1995; VanLehn, 2011). En general, estos enfoques coinciden en algo: es más útil intervenir durante el proceso que limitarse a evaluar el resultado final.

Con la aparición reciente de modelos de lenguaje, este escenario cambió bastante. Hoy es posible interactuar con herramientas que no solo explican código, sino que también pueden proponer soluciones completas a partir de una consigna en lenguaje natural (Chen et al., 2021). Esto abrió nuevas posibilidades, pero también algunas dudas. En la práctica, no siempre queda claro cuánto aprende el estudiante cuando la respuesta está disponible de forma inmediata. Algunos trabajos recientes empiezan a señalar este punto, marcando que el uso sin mediación puede llevar a una participación más pasiva (Kasneci et al., 2023).

A partir de esto, comenzaron a aparecer propuestas que intentan encontrar un punto intermedio. En lugar de evitar estas herramientas, buscan integrarlas de forma más controlada, incorporando guías, restricciones o dinámicas que obliguen al estudiante a involucrarse en la resolución (Wang et al., 2024).

El enfoque que se plantea en esta investigación va en esa dirección. La idea no es usar la inteligencia artificial para resolver los ejercicios, sino para acompañar el proceso ordenando el abordaje del problema desde el principio, proponiendo una forma de trabajo que ayude a estructurar la solución paso a paso.

3 Propuesta

Este trabajo propone un tutor que guía al estudiante en la resolución de problemas, promoviendo el pensamiento analítico y la comprensión progresiva de conceptos.

La Fig. 1 presenta una visión general del flujo de interacción propuesto, incluyendo las distintas etapas del proceso y el rol del tutor basado en inteligencia artificial.

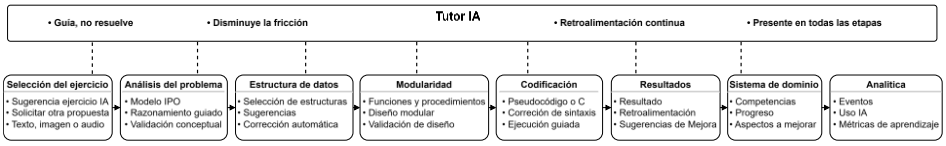

Fig. 1. Flujo pedagógico implementado por Codexi para el aprendizaje guiado de algoritmos. El Tutor de IA acompaña transversalmente todas las etapas del proceso, proporcionando guía y retroalimentación continua, mientras que la propuesta integra la selección del ejercicio, el análisis del problema, las estructuras de datos, la modularización, la codificación, la evaluación de resultados, el sistema de dominio y los mecanismos de analítica. Fuente: Elaboración propia.

Un aspecto central del diseño es la forma en que se utiliza la inteligencia artificial. El tutor actúa como un componente transversal que acompaña cada una de las etapas del proceso de resolución, brindando retroalimentación continua sin resolver directamente los ejercicios. Su intervención se encuentra integrada en un recorrido pedagógico estructurado, orientado a promover el razonamiento progresivo del estudiante mediante preguntas, validaciones conceptuales y sugerencias acordes con la etapa en curso. De esta manera, la asistencia se adapta al avance del alumno y busca reducir la dependencia de respuestas directas, favoreciendo la construcción activa del conocimiento.

A su vez, se incorporaron mecanismos de validación conceptual. Antes de avanzar de nivel, el sistema plantea preguntas breves que permiten verificar si los conceptos principales fueron comprendidos. Estas instancias no buscan ser evaluaciones formales, sino puntos de control que refuercen el proceso de adopción de contenidos.

El sistema también integra indicadores de progreso que reflejan el estado de avance del estudiante. Estos indicadores permiten identificar aspectos consolidados y áreas que requieren mayor práctica, facilitando una lectura rápida del desempeño.

Finalmente, se incluyen elementos de gamificación que introducen incentivos para la continuidad. El avance en los ejercicios otorga puntos que pueden utilizarse, a criterio por parte del estudiante, para solicitar distintos niveles de ayuda dentro de la resolución de ejercicios, incorporando un costo leve a la asistencia directa y promoviendo una participación más activa.

Codexi se encuentra disponible en: https://codexiapp.com/ .

4 Diseño del sistema

La implementación de Codexi se desarrolló como una aplicación web con una arquitectura cliente-servidor. El backend, desarrollado en Python, centraliza la lógica pedagógica, la gestión de ejercicios y la interacción con servicios de inteligencia artificial. El tutor conversacional utiliza el modelo GPT-4o mini de OpenAI para proporcionar asistencia en tiempo real mediante texto e interpretación de imágenes, mientras que las respuestas por voz se procesan mediante Whisper-1 para su transcripción automática.

La etapa de codificación incorpora un compilador integrado que permite validar automáticamente las soluciones desarrolladas tanto en lenguaje C como en el pseudocódigo utilizado en la materia. Para ello, Codexi implementa un conjunto de reglas de transformación que convierten internamente el pseudocódigo a lenguaje C de manera transparente para el estudiante, preservando la notación empleada durante el proceso de aprendizaje y posibilitando su compilación y ejecución automática.

Durante la compilación, el entorno también integra la asistencia del Tutor de IA para el análisis de errores de sintaxis y compilación. En lugar de proponer la corrección completa del código, el sistema genera pistas contextualizadas que orientan al estudiante sobre el origen del error y favorecen su resolución autónoma, manteniendo la estrategia pedagógica de guiar el razonamiento sin proporcionar soluciones directas.

La interacción con el modelo de lenguaje no se realiza de manera directa. Codexi incorpora una capa de orquestación pedagógica que interpreta el estado del ejercicio, la etapa de resolución en la que se encuentra el estudiante y las reglas didácticas definidas para cada instancia. Esta capa determina el tipo de asistencia que puede ofrecer el tutor, regulando las respuestas del modelo para mantener una progresión consistente del proceso de aprendizaje.

Un aspecto central del diseño es la capacidad para ofrecer indicios, reformular planteos y apoyar con preguntas que orientan el razonamiento del estudiante en cada una de las etapas de resolución. Este comportamiento se implementa a través de pautas que restringen la entrega de soluciones completas, priorizando intervenciones que favorecen la comprensión progresiva del problema.

Para el desarrollo práctico, se incorporó la ejecución de código en C dentro de la aplicación, por corresponder al lenguaje de programación utilizado en la asignatura objetivo, permitiendo integrar la plataforma al contexto real de enseñanza sin modificar la metodología de la cátedra. La integración se realizó utilizando un entorno controlado que garantiza la seguridad durante su uso; lo cual permite integrar la práctica directamente sin necesidad de herramientas externas.

Asimismo, se integraron mecanismos de monitoreo que registran eventos de uso y permiten analizar el comportamiento de los usuarios dentro del sistema.

Las decisiones de implementación se orientaron a acompañar la propuesta pedagógica sin complejizar la experiencia, manteniendo un equilibrio entre los procesos internos del sistema y las necesidades del usuario para enfocarse en la adopción de los conocimientos y resolución de ejercicios.

5 Resultados preliminares

La evaluación preliminar se realizó durante el primer cuatrimestre de 2026 con alumnos de dos comisiones de la asignatura Algoritmos y Estructuras de Datos de la UTN. La plataforma fue presentada como una herramienta complementaria y su utilización como voluntaria. Al momento de elaborar este trabajo, registraba 51 usuarios registrados, sobre los cuales se analizaron las métricas de uso obtenidas automáticamente mediante el registro de eventos de la aplicación.

Durante el período analizado se iniciaron 141 ejercicios, de los cuales 91 fueron completados, alcanzando una tasa de finalización del 65 %. El tiempo promedio de resolución fue de 17 min 22 s por ejercicio.

Un primer aspecto a destacar es que la mayoría de los usuarios comienza los ejercicios, pero no siempre los finaliza. Este comportamiento es esperable en una etapa temprana, especialmente considerando que los estudiantes aún se están familiarizando tanto con la dinámica de la plataforma como con los contenidos. Los datos muestran que una parte de los estudiantes no completa los ejercicios iniciados, aspecto que será objeto de análisis en futuras evaluaciones.

Del análisis de las interacciones registradas se observaron indicios de tres patrones de uso. En primer lugar, los estudiantes tendieron a estructurar mejor la comprensión del problema antes de comenzar a programar. En segundo lugar, se observó una menor

dependencia de respuestas completas, con mayor uso de pistas intermedias. Por último, la progresión por niveles favoreció una interacción más sostenida con los ejercicios. Estos comportamientos pueden estar relacionados con el diseño del tutor, particularmente con la secuencia obligatoria de pasos y el uso de retroalimentación incremental.

En conjunto, estos resultados preliminares sugieren que el enfoque propuesto contribuye a promover una participación más activa y una mejor organización del proceso de resolución, aunque aún se requiere mayor evidencia para validar su impacto de forma concluyente.

Este trabajo posee algunas limitaciones que deben ser consideradas. En primer lugar, la cantidad de usuarios evaluados hasta el momento es reducida, lo que limita la posibilidad de generalizar los resultados obtenidos. Asimismo, no se ha realizado aún una comparación experimental controlada con métodos tradicionales de enseñanza, por lo que las conclusiones se basan principalmente en observaciones iniciales de uso. Finalmente, futuras líneas de investigación incluyen la ampliación del conjunto de usuarios, la incorporación de métricas de aprendizaje más robustas y la realización de estudios comparativos que permitan validar con mayor profundidad el impacto del enfoque propuesto.

En síntesis, el enfoque busca reposicionar el rol de la inteligencia artificial en el aprendizaje de algoritmos, no como un generador de respuestas, sino como un facilitador del proceso de razonamiento. Los resultados obtenidos respaldan la viabilidad de integrar modelos de lenguaje dentro de un flujo pedagógico estructurado para acompañar el aprendizaje de algoritmos, constituyendo una alternativa al uso de asistentes conversacionales de propósito general.

References

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. The Journal of the Learning Sciences, 4(2), 167–207.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. de O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, Ł., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Bekman, S., Welch, E., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Carr, A., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., & Zaremba, W. (2021). Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374.

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdinger, F. W., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., Stadler, M., Weller, J., & Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. Learning and Individual Differences, 103, 102274.

VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. Educational Psychologist, 46(4), 197–221.

Wang, Y., Li, H., Zheng, Q., & Zhao, Y. (2024). Exploring guided interaction strategies in AI-assisted programming education. Computers and Education: Artificial Intelligence, 6, 100198.