Artículo de Revisión

Impacto de modelos de lenguaje de gran tamaño en calidad y eficiencia de generación de código: Revisión Sistemática de Literatura

Impact of large language models on quality and efficiency of code generation: Systematic Literature Review

Viviana Fernanda Quevedo-Tumailli[1], Sandra Elizabeth Arias Calderón[1], Verónica Alexandra Ortega Manjarrez[1], Bernabe Ortega-Tenezaca[1]

[1]*Universidad Estatal Amazónica, Puyo, Ecuador, 16101;*

se.ariasc@uea.edu.ec; va.ortegam@uea.edu.ec; bortega@uea.edu.ec

*Correspondencia: vquevedo@uea.edu.ec

Citación: Quevedo-Tumailli, V.; Arias, S.; Ortega, V. & Ortega-Tenezaca, B., (2025). Impacto de modelos de lenguaje de gran tamaño en calidad y eficiencia de generación de código: Revisión Sistemática de Literatura. *Novasinergia. 8*(1). 52-66.

https://doi.org/10.37135/ns.01.15.10

Recibido: 28 octubre 2024
Aceptado: 02 diciembre 2024
Publicado: 08 enero 2025

Novasinergia
ISSN: 2631-2654

Resumen: Los Modelos de Lenguaje de Gran Tamaño (LLM) están revolucionando la eficiencia y automatización en la generación de código de programación. Este trabajo tiene como objetivo proporcionar una visión general del estado del arte de estos modelos y su aplicación específica en la programación. La metodología permitió identificar 549 publicaciones iniciales, cribadas en fases sucesivas hasta seleccionar 27 artículos clave para el análisis cualitativo. Los resultados muestran que los LLM mejoran significativamente la calidad del código generado, optimizan la productividad de los programadores y apoyan el aprendizaje mediante herramientas como ChatGPT, Codex y GitHub Copilot. El análisis destaca que el 66,7% de los artículos revisados se publicaron en el 2024, reflejando el creciente interés en este campo. Estados Unidos lidera la investigación con 11 publicaciones, seguido de Países Bajos y Suiza. Los estudios seleccionados se filtraron para generar una síntesis crítica que evalúa la calidad y aplicabilidad de los LLM en la escritura y depuración de código. Esta revisión analiza las principales ventajas de los LLM, como la mejora en la productividad y la generación de código optimizado. Sin embargo, también aborda desafíos como la precisión y el riesgo de errores o código no seguro. Los LLM pueden mejorar significativamente la calidad del código generado. También pueden facilitar la curva de aprendizaje de los nuevos programadores. No obstante, se resalta la necesidad de mantener una supervisión humana rigurosa para garantizar la fiabilidad de las soluciones generadas por los modelos.

Palabras clave: Editores de código, Generación de código, Metodología de revisión, Modelos de Lenguaje de Gran Tamaño, Revisión sistemática de Literatura.

Abstract: *Large Language Models (LLMs) are revolutionizing efficiency and automation in programming code generation. This study aims to provide an overview of these models' state-of-the-art and specific programming applications. The methodology enabled the identification of 549 initial publications, screened through successive phases to select 27 key articles for qualitative analysis. The results show that LLMs significantly improve the quality of generated code, enhance programmer productivity, and support learning through tools such as ChatGPT, Codex, and GitHub Copilot. The analysis highlights that 66.7% of the reviewed articles were published in 2024, reflecting a growing interest in this field. The United States leads research with 11 publications, followed by the Netherlands and Switzerland. The selected studies were filtered to generate a critical synthesis evaluating the quality and applicability of LLMs in code writing and debugging. This review examines the main advantages of LLMs, such as increased productivity and optimized code generation. However, it also addresses challenges like accuracy issues and the risk of generating errors or insecure code. LLMs can significantly enhance generated code quality and facilitate new programmers' learning curve. Nevertheless, the need for rigorous human oversight is emphasized to ensure the reliability of the solutions generated by these models.*

Keywords*: Code Editors, Code Generation, Review Methodology, Large Language Models, Systematic Literature Review.*

1. Introducción

Los Modelos de Lenguaje de Gran Tamaño (LLM, por sus siglas en inglés) representan uno de los avances más significativos en el campo del Procesamiento del Lenguaje Natural (NLP, por sus siglas en inglés) y han demostrado ser herramientas poderosas en una variedad de tareas que incluyen generación de texto, traducción automática, y generación de código de programación. Estos modelos, entrenados en enormes cantidades de datos textuales, tienen la capacidad de generar contenido coherente y altamente contextualizado, lo que los ha convertido en una tecnología clave en el desarrollo de software asistido por la Inteligencia Artificial (AI, por sus siglas en inglés) (Kau et al., 2024).

La arquitectura transformer introducida por Vaswani es la base de los LLM modernos (Vaswani et al., 2017). Este tipo de arquitectura utiliza mecanismos de autoatención que permiten a los modelos procesar y generar secuencias de texto de manera más eficiente que las arquitecturas anteriores, como las redes neuronales recurrentes.

Los LLM, como el Transformador Generativo Preentrenado (GPT, por sus siglas en inglés) y sus sucesivos desarrollos, han sido adaptados específicamente para mejorar la eficiencia en la generación de código, mostrando una notable habilidad para completar código, generar soluciones a problemas de programación y entender consultas complejas en lenguaje natural relacionadas con la programación. Weber (2024) resaltan cómo estos modelos no solo mejoran la productividad de los programadores, sino que también cambian radicalmente sus métodos de trabajo al ofrecer soluciones de codificación en tiempo real.

El uso de LLM en la programación se ha popularizado en los últimos años gracias a herramientas como GitHub Copilot, que utiliza Codex para generar fragmentos de código a partir de descripciones en lenguaje natural. Esta capacidad de los LLM ha transformado la forma en que los desarrolladores interactúan con los entornos de programación, permitiéndoles generar soluciones rápidas y eficientes sin necesidad de escribir cada línea de código manualmente (Chen et al., 2021).

Con el rápido avance en el desarrollo de los LLM y su integración en los editores de código, es fundamental analizar cuidadosamente como estas herramientas están influyendo en el desarrollo de software especialmente en la creación de código de programación. Esta revisión busca identificar las fortalezas y debilidades de los modelos actuales, brindando una visión más clara de cómo pueden mejorar la productividad de los programadores y facilitar el proceso de desarrollo de software. Además, esta investigación justifica su relevancia al abordar la creciente adopción de herramientas basadas en IA en el ámbito profesional, donde la eficiencia y la precisión son esenciales.

Los estudios recientes sobre la aplicación de LLM en la generación de código muestran resultados mixtos. Por ejemplo, exploran el impacto de estos modelos en la educación en desarrollo de software, encontrando que una dependencia excesiva de LLM puede disminuir el rendimiento académico (Jošt et al., 2024). Por otro lado, describen cómo las técnicas avanzadas pueden combinar la generación de programas LLM con métodos de programación genética para mejorar la síntesis de programas (Tao et al., 2024). Sin embargo, investigaciones recientes también subrayan algunos desafíos, como la generación de código

incorrecto o inseguro, lo que obliga a los programadores a revisar manualmente las sugerencias generadas por IA (Chen et al., 2021). Estas investigaciones ilustran el amplio espectro de aplicaciones, beneficios y limitaciones que poseen los LLM en contextos de programación.

Un análisis crítico revela que, aunque los LLM ofrecen una mejora significativa en términos de velocidad y eficiencia, aún existe una dependencia considerable de la supervisión humana para garantizar la calidad y seguridad del código generado (Bender et al., 2021). La transparencia y reproducibilidad de los resultados son aspectos que requieren más investigación, ya que los LLM pueden generar resultados diferentes a partir del mismo conjunto de entradas debido a su naturaleza estocástica (Fritsch & Jatowt, 2024).

Con el rápido avance y la creciente popularidad de los LLM en la generación de código, esta revisión sistemática de la literatura se propone analizar en profundidad la manera en que estos modelos están redefiniendo el desarrollo del software. El objetivo principal es investigar cómo influyen los LLM en la calidad y eficiencia del código generado por programadores y estudiantes. Para ello, se considerarán estudios de caso y evaluaciones empíricas para proporcionar una visión completa y actualizada sobre este campo. Esta revisión busca sintetizar el conocimiento existente, identificar deficiencias en la investigación actual, y sugerir direcciones para futuros desarrollos y estudios en el área.

2. Metodología

Este estudio combina análisis cualitativos y cuantitativos para investigar la utilización de los LLM en la generación de código y evaluar su impacto. Con un enfoque exploratorio y descriptivo, la investigación se centró en revisar, gestionar y sistematizar publicaciones en un campo de estudio que sigue evolucionando rápidamente. Se incluyeron estudios de los últimos 5 años que abordaron el uso de LLM en programación, con criterios de inclusión y exclusión, asegurando la relevancia y calidad de las fuentes. Se utilizó el formato de una Revisión Sistemática de la Literatura (SLR) que describe el proceso desde la definición de las preguntas de investigación hasta la elaboración del artículo, incluyendo la selección y evaluación de documentos. Los datos se recolectaron de bases académicas, utilizando filtros específicos para identificar estudios relevantes del período analizado. La metodología se basó en el modelo de la declaración PRISMA traducido al español como "Elementos de Reporte Preferidos para Revisiones Sistemáticas y Metaanálisis" (Page et al., 2021).

En la primera fase llamada Identificación de PRISMA se definieron las bases de datos más relevante y accesibles como Scopus y Google Scholar. Los tipos de documentos aceptables fueron únicamente artículos científicos, los idiomas analizados fueron español e inglés, y se consideraron publicaciones de los últimos 5 años (2020 – 2024). Aunque el rango explícito incluye hasta 2024, se incluyeron documentos publicados en 2025 que estuvieran disponibles al momento de la revisión para asegurar la actualidad de los resultados. Una particularidad en Scopus fue limitar los resultados en las áreas: Ciencias de la Computación e Ingeniería. Otro punto muy importante fue la utilización de la herramienta Publish or Perish (Harzing, 2007) para encontrar las publicaciones en Google Scholar. Es importante reconocer la importancia de Scopus como una base de datos consolidada para análisis

académicos y fue utilizada como fuente principal para la identificación de publicaciones relevantes. Sin embargo, se consideró necesario incluir Google Scholar como complemento de Scopus porque es una herramienta ampliamente utilizada por la comunidad académica debido a su accesibilidad gratuita, esto permitió obtener literatura relevante en áreas emergentes como LLM. Los documentos identificados responden a la búsqueda de la siguiente fórmula: (“Large Language Models" AND "code learning”). En total, se identificaron 549 publicaciones (404 en Scopus y 145 en Google Scholar) y se descargaron en el gestor documental Zotero, a una base de datos compartida, usada para la revisión. Seguidamente, se agruparon en un solo listado y se eliminó 1 publicación duplicada.

Para el desarrollo del presente trabajo se seleccionaron dos términos de búsqueda “Large Language Models” y “code learning”. Es importante mencionar que el primer término no ha sido formalizado como un descriptor porque pertenece a un dominio de rápida evolución (tecnología e inteligencia artificial), donde los términos cambian constantemente. Aunque existe vocabulario relacionado en tesauros (como “natural language processing” o “machine learning”), “Large Languaje Models” es una subcategoría dentro de estos conceptos generales, y los tesauros tienden a priorizar términos más amplios y menos específicos. El segundo término no es un descriptor formal porque cruza dos áreas: Programación informática y pedagogía. El término “Code learning” es una expresión popularizada en literatura técnica y educativa reciente, especialmente en artículos que discuten la enseñanza de la programación y el uso de herramientas automatizadas como LLM. Sin embargo, no ha alcanzado suficiente reconocimiento formal en la comunidad científica para ser incluido en tesauros.

En la segunda fase llamada Cribado, se revisaron 548 títulos y abstracts aplicando los criterios de inclusión y exclusión de las publicaciones para cribar los resultados.

Criterios de exclusión (CExn):

- CEx1: Se excluyeron las publicaciones que no estaban relacionadas con la temática investigada sobre LLM en la generación de código.
- CEx2: Se excluyeron las publicaciones que no ponían el foco en la generación de código de programación excluyendo temas relacionados con otros aspectos, como el procesamiento de lenguaje natural en general o aplicaciones en otras áreas que no son relevantes.
- CEx3: Se excluyeron las publicaciones que no realizaban una nueva contribución, ya que solo se buscaron investigaciones originales o análisis únicos que contribuyan directamente al campo.
- CEx4: Se excluyeron las publicaciones a las que no se tuvo acceso completo, lo que limitó la posibilidad de evaluar en detalle la metodología y los resultados afectando la calidad de la revisión sistemática.
- CEx5: Se excluyeron las publicaciones que no estaban en idioma inglés o español debido a las habilidades lingüísticas del grupo de investigación.
- CEx6: Se excluyeron las publicaciones que eran preprints, ya que esto afectaba la calidad y confiabilidad de los datos y conclusiones presentados al no pasar por un proceso de revisión por pares.

- CEx7: Se eliminaron estudios que no proporcionaban evidencia sólida, que no estaban centrados en la mejora de código mediante LLM o presentaban análisis insuficientes, para garantizar la calidad y relevancia de los datos analizados.

Criterios de inclusión (CInn):

- CIn1: Se incluyeron las publicaciones que se centraban en el aprendizaje de la programación para comprender cómo los LLM apoyan a los programadores.
- CIn2: Se incluyeron las publicaciones que aportaban comparaciones entre diferentes modelos LLM, permitiendo evaluar diferencias para comprender las contribuciones específicas de los LLM en la programación.
- CIn3: Se incluyeron las publicaciones que estudiaban modelos LLM, proporcionando una base teórica y técnica para el tema de investigación.
- CIn4: Se incluyeron las publicaciones que aportaban revisiones exhaustivas, análisis cualitativos o datos empíricos sobre el uso de LLM para mejorar la calidad y eficiencia del código generado, asegurando un aporte a la revisión sistemática.

En esta fase se revisaron los títulos de las 548 publicaciones cribadas, cumpliendo los criterios de inclusión (CInn) y de exclusión (CExn) que se establecieron para el estudio, esto permitió enfocar el análisis en los trabajos más relevantes para la investigación. Seguidamente, se evaluaron los abstracts aplicando nuevamente los criterios ya mencionados. En esta segunda revisión se eliminaron 490 publicaciones. En la Figura 1, se resume el procedimiento de selección de las publicaciones.

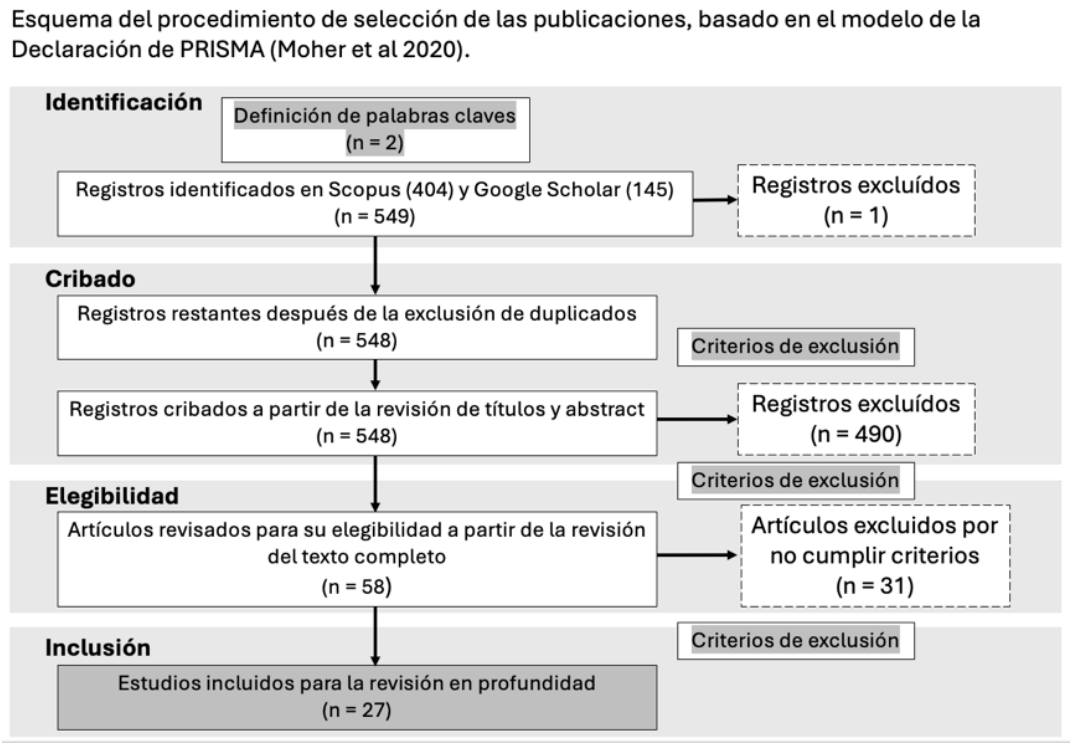

Figura 1: Diagrama de Flujo de fases según modelo PRISMA 2020 (Haddaway et al., 2022).

En la tercera fase llamada Elegibilidad de PRISMA, se leyeron por completo 58 publicaciones para una evaluación en el proceso de determinación de su elegibilidad usando nuevamente los criterios CEx7 y CIn4 en este caso. Se eliminaron 31 publicaciones, quedando 27 documentos para el análisis cualitativo de las contribuciones.

Finalmente, en la última fase llamada Inclusión de PRISMA se registraron los 27 artículos como resultado final, cumplieron con todos los criterios establecidos y respondieron directamente a las preguntas de investigación. En la Figura 2, se muestran los resultados que fueron evaluados en el proceso de selección de los artículos científicos.

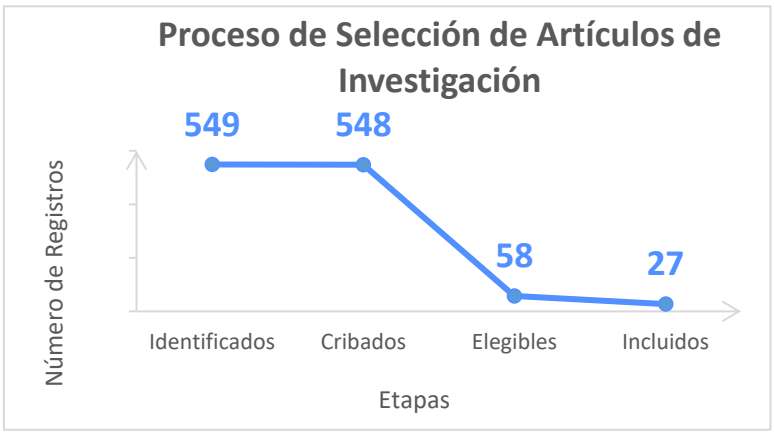

Figura 2: Resultados del proceso de selección.

Se concluyó que se siguió el modelo de PRISMA para realizar la revisión sistemática de literatura. Este proceso incluyó cuatro etapas: Identificación, cribado, elegibilidad e inclusión. Se utilizó la estrategia de búsqueda que se detalla en la Tabla 1.

Tabla 1: Estrategia de búsqueda y selección.

Elementos	Criterios
Palabras clave	*"Large Language Models"*, "*code learning*"
Bases de datos	Scopus, Google Scholar
Filtros aplicados	Publicaciones entre 2020 al 2025, idioma solo en inglés, tipo de documento solo artículos, área temática en Ciencias de la Computación e Ingeniería, artículos revisados por pares
Operadores booleanos	AND

El objetivo planteado en esta revisión sistemática fue investigar cómo influyen los LLM en la calidad y eficiencia del código generado por programadores y estudiantes según lo documentado en la literatura científica. Para alcanzar dicho objetivo se planteó preguntas de investigación basadas en el modelo WWH (Foster & Jewell, 2017):

- P1: ¿Qué efecto tienen los LLM en la calidad del código generado por programadores y estudiantes?
- P2: ¿Por qué es importante estudiar la influencia de los LLM en la calidad y eficiencia del código en el contexto de la programación y el aprendizaje?"
- P3: ¿Cómo impactan los LLM en los aspectos específicos de la eficiencia y la calidad del código, y cómo se miden estos efectos en la literatura científica?

En cada una de las fases del modelo de la declaración PRISMA, los criterios se aplicaron rigurosamente para garantizar que solo los estudios más relevantes, de calidad y alineados con la pregunta de investigación formen parte del análisis final.

En la Tabla 2 se encuentran las referencias incluidas en el análisis. Se incluyen dos revistas de Austria, once de Estados Unidos, cinco de los Países Bajos, tres de Reino Unido, cinco de Suiza y uno de España, mostrando una notable representación en la Unión Europea y Estados Unidos en publicaciones. En cuanto a metodología, veinte artículos son cuantitativos, tres cualitativos y cuatro mixtos. También se muestra la mayor concentración de publicaciones en el último año 2024.

Tabla 2: Datos identificativos.

Nro.	Autores	Año	País	Método
1	Yang, Liu & Yin	2021	Suiza	Mixto
2	Kovačević et al.	2022	Países Bajos	Cuantitativo
3	Azaiz, Deckarm & Strickroth	2023	Austria	Mixto
4	Rahman & Watanobe	2023	Suiza	Mixto
5	Kazemitabaar et al.	2023	Estados Unidos	Cuantitativo
6	Matei-Dan	2023	Reino Unido	Cuantitativo
7	Wang et al.	2023	Estados Unidos	Cuantitativo
8	Robledo et al.	2023	España	Cuantitativo
9	Yun et al.	2024	Países Bajos	Cuantitativo
10	Bae, Kwon & Myeong	2024	Suiza	Cuantitativo
11	Zhang et al.	2024	Estados Unidos	Cuantitativo
12	Fitero-Dominguez et al.	2024	Reino Unido	Cuantitativo
13	Jošt, Taneski & Karakatič	2024	Suiza	Cuantitativo
14	Hao, Shi & Liu	2024	Suiza	Cuantitativo
15	Pornprasit & Tantithamthavorn	2024	Países Bajos	Cuantitativo
16	Echeverría et al.	2024	Estados Unidos	Cuantitativo
17	Wan et al.	2024	Estados Unidos	Cuantitativo
18	Li et al.	2024	Reino Unido	Cuantitativo
19	Haindl & Weinberger	2024	Austria	Mixto
20	Deineha	2024	Estados Unidos	Cuantitativo
21	Rocha et al.	2024	Estados Unidos	Cualitativo
22	Zhou et al.	2024	Estados Unidos	Cuantitativo
23	Tu et al.	2024	Estados Unidos	Cuantitativo
24	Champa et al.	2024	Estados Unidos	Cuantitativo
25	Rongcun et al.	2024	Países Bajos	Cuantitativo
26	Talib et al.	2024	Estados Unidos	Cualitativo
27	Zhou et al.	2025	Países Bajos	Cualitativo

3. Resultados

En este trabajo, se llevó a cabo una búsqueda exhaustiva de los artículos científicos utilizando las bases de datos de prestigio y accesibles como Google Scholar y Scopus, aplicando un SLR, en inglés y español en los últimos 5 años. Sin embargo, los resultados

obtenidos fueron solo en inglés. Se evaluaron los títulos y resúmenes para seleccionar los estudios que se ajusten a la pregunta de investigación, es decir, aquellos que traten específicamente sobre la influencia de los LLM en la calidad y eficiencia del código. Los términos de búsqueda incluyeron " Large Language Models " y " code learning ". Se excluyeron aquellos artículos que no estaban relacionados directamente con el aprendizaje o generación de código o que no trataban el tema de modelado del lenguaje. Se excluyeron los temas que no estaban relacionados con el uso de LLM en la enseñanza o aprendizaje de la programación, aunque mencionaran LLM en otros contextos como la biología, procesamiento de imágenes o aplicaciones médicas. Se excluyeron estudios que no proporcionaban una evaluación empírica, crítica o estructurada sobre el uso de LLM para el aprendizaje de programación, o aquellos que no ofrecían datos cuantitativos o cualitativos que pudieran contribuir a la evaluación de su efectividad.

Luego de la aplicación del modelo de la declaración PRISMA, la Figura 3 proporciona información sobre los porcentajes de publicaciones, se obtiene el 74,1% de publicaciones consultadas en Scopus y el 25,9% en Google Scholar. La Figura 4 indica la cantidad y el porcentaje de publicaciones seleccionadas por años como resultado el porcentaje de artículos en ambas bases.

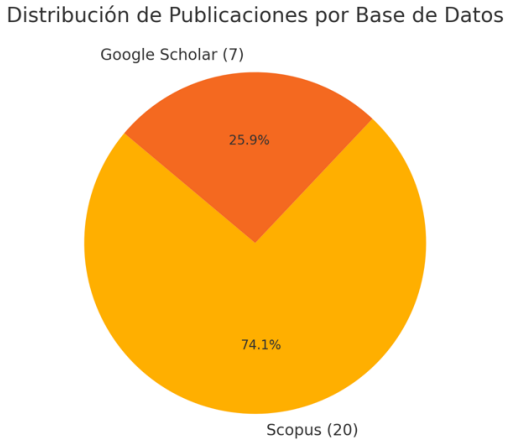

Figura 3: Publicaciones en Scopus y Google Scholar.

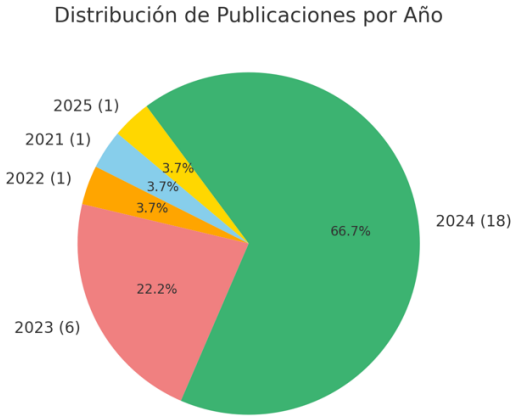

Figura 4: Publicaciones por años.

La Figura 5 ilustra la distribución de artículos científicos por país relacionados con la investigación enfocándose en los países que lideran la producción de literatura científica en este ámbito, reflejando la dinámica geográfica de la investigación en inteligencia artificial y educación tecnológica.

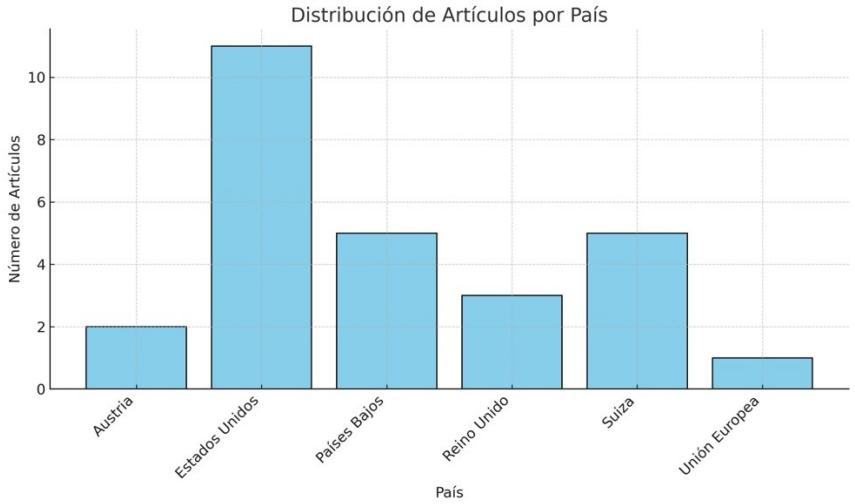

Figura 5: Publicaciones por paises.

La Tabla 3 presenta un resumen por categorías de las palabras clave obtenidas de los artículos revisados. Las palabras fueron organizadas en seis categorías principales, reflejando los temas más relevantes relacionados con "Large Languaje Models" y "code learning". Estas categorías reflejan las áreas temáticas recurrentes en los estudios analizados.

Tabla 3: Frecuencia de Palabras recurrentes agrupadas.

Nro.	Categoría	Frecuencia	Palabras recurrentes
1	Modelos de AI	19	large language models, mistral, GPT, Copilot, ChatGPT, code llama, claude-3.5
2	Herramientas de Programación	6	Stack overflow post, React, GitHub Copilot, GitHub discussions, GitHub issues
3	Educación en Programación	6	Programming education, AI for programming education
4	Reparación de código	4	Debugging, vulnerability repair, automated program repair
5	Seguridad	3	Software security, vulnerability detection, information security
6	Técnicas	3	Deep learning, prompt learning, prompt engineering

La Tabla 4 proporciona un resumen de los resultados de las investigaciones seleccionadas, enfocándose en cómo los LLM influyen en la calidad y eficiencia del código generado por programadores y estudiantes.

Tabla 4: Resumen de resultados de las investigaciones seleccionadas.

Autores	Principales Ideas
Yang, Liu & Yin	Investigación sobre el impacto de NLSCG en generación de código a partir de descripciones de lenguaje natural.
Kovačević et al.	Comparación de detección de 'code smells' usando ML y heurísticas tradicionales.
Azaiz, Deckarm & Strickroth	Investigación del uso de IA para retroalimentación personalizada en programación.

Rahman & Watanobe	Exploración de oportunidades y amenazas de ChatGPT en la educación y aprendizaje de programación.
Kazemitabaar et al.	Estudio sobre el impacto de herramientas de IA en tareas introductorias de programación con jóvenes de 10 a 17 años.
Matei-Dan	Evaluación de ChatGPT como herramienta de revisión de código en cursos de programación.
Wang et al.	Propuesta de ChatDANCE para mejorar la búsqueda de código mediante datos aumentados y filtrado de baja calidad.
Robledo et al.	Resumen del proyecto DECODER, que integra NLP en herramientas de software para detección de bugs y documentación automática.
Yun et al.	Se analiza la eficacia de LLMs para la generación de resúmenes específicos de proyectos, proponiendo P-CodeSum para mejorar esta tarea.
Bae, Kwon & Myeong	Evaluación de modelos LLM en la detección de vulnerabilidades en el código con diferentes técnicas de prompting.
Zhang et al.	Uso de LLMs en reparación automática de programas (APR) en educación, propuesta de PyDex para reparación de errores en asignaciones de Python.
Fitero-Dominguez et al.	Nueva representación de modificaciones de código usando LLMs para mejorar la reparación automática de vulnerabilidades.
Jošt, Taneski & Karakatič	Estudio sobre el impacto negativo del uso excesivo de LLMs en el aprendizaje de programación en estudiantes.
Hao, Shi & Liu	Mejoras en APR mediante el aprendizaje basado en currículo y aumento de código en PLMCs.
Pornprasit & Tantithamthavorn	Investigación sobre el uso de LLMs para automatizar la revisión de código mediante ajustes de fine-tuning y prompting.
Echeverría et al.	Evaluación del rendimiento de ChatGPT en la resolución de exámenes de ingeniería de software.
Wan et al.	Propuesta de CEMR para reparar errores en programas de estudiantes mediante el uso de modelos de error contextual.
Li et al.	Desarrollo de FSATD, que combina LLMs y modelos pequeños para detección de deudas técnicas en código.
Haindl & Weinberger	Estudio sobre experiencias de estudiantes usando ChatGPT en un curso de Java, explorando su utilidad para aprendizaje de conceptos de programación.
Deineha	Aplicación de LLMs y ML para optimización de compiladores en Lambda Calculus.
Rocha et al.	Evaluación del uso de NLP para apoyo en aprendizaje de microcontroladores en cursos de ingeniería.
Zhou et al.	Propuesta de LLM4PatchCorrect para evaluar la corrección de parches generados por herramientas APR.
Tu et al.	Uso de LLMs para aislamiento de bugs en compiladores mediante el enfoque LLM4CBI.
Champa et al.	Análisis del uso de ChatGPT en el apoyo a tareas de desarrollo de software, investigando efectividad y calidad de asistencia.
Rongcun et al.	Propuesta de SCL-CVD para mejorar la detección de vulnerabilidades en código usando aprendizaje contrastivo supervisado.
Talib et al.	Estudio sobre las experiencias de programadores con herramientas de generación de código AI como ChatGPT.
Zhou et al.	Estudio sobre problemas comunes de GitHub Copilot y soluciones potenciales, destacando cuestiones de operación y compatibilidad.

4. Discusión

El análisis de los artículos seleccionados converge en la exploración de los LLM para tareas de desarrollo de software, automatización y apoyo educativo. Los hallazgos cubren una amplia gama de aplicaciones, desde la generación de resúmenes específicos de proyectos y la reparación automática de errores en código, hasta la evaluación de exámenes y la detección de vulnerabilidades.

La distribución temporal destaca que el 2024 es el año con la mayor cantidad de publicaciones con un total de 18 artículos, representando un 66,7% del total. Este incremento refleja un interés sostenido y un aumento en la investigación sobre los LLM, posiblemente por la popularización de herramientas como ChatGPT y otros modelos avanzados. En el año 2023 con 6 artículos representando un 22,2%, muestra un crecimiento respecto a años anteriores, consolidando la relevancia de los LLM en la investigación científica. Los años 2021 y 2022, cada uno con 1 publicación con el 3,7%, representa el inicio de la exploración del tema, cuando los LLM comenzaban a implementarse en aplicaciones prácticas. La proyección hacia el 2025, con un artículo identificado, sugiere que esta área de investigación seguirá siendo relevante en los próximos años.

Los datos de la figura 5 revelan que Estados Unidos es el principal contribuyente en la producción de artículos sobre LLM y aprendizaje de programación con un total de 11 publicaciones. Esto refleja su posición como líder mundial en innovación tecnológica, investigación en AI y educación en ciencias computacionales, respaldadas por instituciones de renombre y empresas como OpenAI, Google y Microsoft. Paises Bajos y Suiza con 5 publicaciones cada uno posiblemente debido a su fuerte inversión en investigación y desarrollo. Reino Unido sigue con 3 artículos, lo que subraya su enfoque en el uso de tecnologías emergentes en la educación y la investigación académica. Por otro lado, paises como Australia y Unión Europea contribuyen con menos publicaciones.

La categoría más destacada según la Tabla 3 fue Modelos de AI con una frecuencia de 19 menciones sobre tecnologías y herramientas de AI relevantes para la programación. Otras categorías como Herramientas de Programación y educación en programación también mostraron un interés significativo, con seis menciones cada una, refiriéndose a plataformas y recursos utilizados para desarrollo colaborativo y programación y también sobre aspectos pedagógicos y educativos asociados al uso de LLM. En este punto es importante mencionar herramientas como GitHub Copilot y conceptos como la Ingeniería de Prompt que destacan tecnologías clave en el apoyo a los desarrolladores.

A continuación, se describen los principales resultados en relación con las tres preguntas de investigación planteadas en la sección de metodología.

Los LLM están mostrando un impacto notable en la calidad del código que producen tanto programadores como estudiantes. En el ámbito educativo, herramientas como ChatGPT y Codex están siendo un valioso apoyo para los estudiantes, ya que les ayudan a comprender conceptos de programación y corregir errores de código en tiempo real. Esto permite que se concentren más en el diseño y la lógica, sin quedar atrapados en errores menores de sintaxis (Haindl & Weinberger, 2024). En el desarrollo profesional, herramientas como GitHub Copilot ofrecen sugerencias en tiempo real que pueden facilitar el trabajo de los

programadores. Sin embargo, también pueden presentar problemas de compatibilidad y generar errores ocasionales. Estos problemas resaltan la importancia de una supervisión cuidadosa para asegurar que el código sea de calidad (Zhou et al., 2025). En el caso de los programadores novatos, el uso de herramientas como Codex ha demostrado mejorar la tasa de éxito en la creación de código, aunque el uso excesivo de estas herramientas puede limitar el desarrollo de habilidades independientes (Kazemitabaar et al., 2023). En resumen, los LLM pueden mejorar la calidad del código al reducir errores comunes, pero para mantener la autonomía del programador, es importante equilibrar su uso con supervisión.

Analizar la influencia de los LLM en la calidad y eficiencia del código resulta esencial, dado el potencial de estas herramientas para transformar tanto la educación en programación como el desarrollo de software. La automatización de tareas de revisión y generación de código puede reducir significativamente el tiempo necesario para crear código funcional, lo cual es especialmente importante en contextos donde la productividad es crítica (Pornprasit & Tantithamthavorn, 2024). Sin embargo, surge la preocupación de que los estudiantes lleguen a depender demasiado de estas herramientas, limitando el desarrollo de habilidades críticas en pensamiento y resolución de problemas (Jošt et al., 2024). Además, los LLM también contribuyen a mejorar la precisión en la reparación de errores y detección de vulnerabilidades, apoyando así la seguridad del software (Bae et al., 2024; De Fitero-Dominguez et al., 2024). Por tanto, es fundamental comprender cómo afectan estos modelos tanto la calidad y eficiencia del código como el proceso de aprendizaje y el desarrollo de competencias clave. Esto ayudará a asegurar que su implementación sea tanto efectiva como ética.

Los LLM mejoran la eficiencia y la calidad del código mediante la automatización de tareas como la creación de resúmenes de código, la reparación de errores y la detección de vulnerabilidades. Por ejemplo, herramientas como P-CodeSum han optimizado la creación de resúmenes específicos de proyectos, lo cual facilita la comprensión del código en equipos de trabajo (Yun et al., 2024). En términos de seguridad, modelos avanzados como GPT-4o han mostrado avances significativos en la detección de vulnerabilidades mediante técnicas específicas de prompting, lo cual incrementa tanto la eficiencia como la precisión de los diagnósticos (Bae et al., 2024). La eficiencia de estas herramientas suele medirse con métricas específicas, como la tasa de aciertos (F1 score), precisión, y comparaciones de rendimiento frente a métodos tradicionales o basados en heurísticas (Wang et al., 2024). Estas evaluaciones se basan en estudios cuantitativos y experimentos controlados que analizan la precisión y la reducción de errores, ofreciendo un marco claro para evaluar cómo impactan los LLM en la calidad del código en la literatura científica.

5. Conclusiones

Esta revisión sistemática demuestra que los LLM están transformando el desarrollo de software, la educación en programación y la seguridad informática. En respuesta a las preguntas de investigación, esta revisión sistemática demuestra que los LLM tienen un impacto significativo en la calidad y eficiencia del código generado, tanto en contextos educativos como profesionales. Los LLM han mejorado la calidad del código al ayudar a programadores y estudiantes a reducir errores comunes y comprender conceptos complejos

apoyando especialmente a los novatos mejorando su éxito en la creación de código y permitiendo un enfoque más profundo en el diseño y la lógica. En cuanto al segundo parámetro como es la influencia, los LLM permiten garantizar que su implementación en programación sea efectiva y ética. Además de acelerar la generación de código funcional, también mejora la precisión de la detección de errores y vulnerabilidades, sin embargo, se plantean desafíos, como la posibilidad de inhibir el desarrollo del pensamiento crítico y las competencias de resolución de problemas de los estudiantes, subrayando la necesidad de un enfoque reflexivo para integrar los LLM en la educación y el desarrollo profesional. Por último, con respecto al impacto en la eficiencia y calidad del código, los LLM optimizan tareas como la reparación de errores, la detección de vulnerabilidades y la creación de resúmenes del código, aumentando tanto la eficiencia como la precisión del trabajo en programación. Sin embargo, los resultados también destacan la necesidad de investigaciones adicionales para explorar cómo se puede maximizar su uso sin comprometer la calidad del aprendizaje y la autonomía del programador. Por otro lado, la investigación muestra como la importancia en los LLM ha crecido rápidamente en regiones tecnológicamente avanzadas como Estados Unidos. Se destaca que el 2024 es el año con la mayor cantidad de publicaciones lo que demuestra una creciente accesibilidad de herramientas basadas en LLM. Los LLM han demostrado una gran habilidad para adaptar tareas de programación generando resúmenes y personalizando tareas como la reparación de errores y la optimización de código. Esto abre nuevas posibilidades para desarrollar sistemas de soporte en proyectos grandes, sin embargo, podría en futuras investigaciones buscar el perfeccionamiento de los LLM en áreas específicas y cómo mejorar su precisión en entornos con alta variabilidad. Los modelos como GPT y técnicas avanzadas de aprendizaje contrastivo supervisado han logrado avances en la detección y corrección automática de vulnerabilidades en código. Esto permitirá en futuros estudios la integración de los LLM en sistemas de seguridad que operan en tiempo real. Los modelos como ChatGPT están siendo implementados en entornos educativos, facilitando el aprendizaje y el soporte en tiempo real, pero el riesgo es que una dependencia excesiva afecte el desarrollo de habilidades de pensamiento crítico. Las investigaciones futuras en este punto podrían enfocarse en estrategias que permitan a los LLM potenciar el aprendizaje autónomo, como, por ejemplo, combinando la inteligencia artificial con métodos tradicionales de enseñanza. Por último, los LLM tienen la capacidad para automatizar revisiones de código y detectar “code smells” mejorando tanto la calidad como la eficiencia en el desarrollo de software. En este sentido, los LLM pueden entrenarse para adaptarse a estilos y estándares específicos de cada organización. El uso de los LLM tiene un impacto inmediato y práctico, mejorando la eficiencia y precisión en tareas esenciales de programación con la generación de código, la revisión y la detección de errores. En el ámbito educativo, pueden transformar la enseñanza de la programación al hacerla más accesible y adaptada a cada estudiante. Sin embargo, se debe considerar los desafíos técnicos y éticos, especialmente para evitar una dependencia excesiva que limite el desarrollo de habilidades críticas en los estudiantes.

Contribuciones de los autores

En concordancia con la taxonomía establecida internacionalmente para la asignación de créditos a autores de artículos científicos (https://casrai.org/credit/). Los autores declaran sus contribuciones en la siguiente matriz:

	Quevedo-Tumailli, V.	Arias, S.	Ortega, V.	Ortega-Tenezaca, B.
Conceptualización	■			
Análisis formal				■
Investigación	■			■
Metodología	■			
Recursos		■	■	
Validación		■		■
Redacción – revisión y edición	■			

Conflicto de Interés

Los autores declaran que no existen conflictos de interés de naturaleza alguna con la presente investigación.

Referencias

Bae, J., Kwon, S., & Myeong, S. (2024). Enhancing Software Code Vulnerability Detection Using GPT-4o and Claude-3.5 Sonnet: A Study on Prompt Engineering Techniques. *Electronics*, *13*(13), 2657. https://doi.org/10.3390/electronics13132657

Bender, E. M, Gebru T., McMillan-Major A., Shmitchell S. (2021). *On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?*. In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT '21). Association for Computing Machinery, New York, NY, USA, 610–623. https://doi.org/10.1145/3442188.3445922

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., … Zaremba, W. (2021). Evaluating Large Language Models Trained on Code. *arXiv Computer Science*. https://doi.org/10.48550/arXiv.2107.03374

De Fitero-Dominguez, D., Garcia-Lopez, E., Garcia-Cabot, A., & Martinez-Herraiz, J.-J. (2024). Enhanced automated code vulnerability repair using large language models. *Engineering Applications of Artificial Intelligence*, *138*. https://doi.org/10.1016/j.engappai.2024.109291

Foster, M. J., & Jewell, S. T. (2017). *Assembling the pieces of a systematic review: A guide for librarians*. Rowman & Littlefield Publishers.

Fritsch, R. F., & Jatowt, A. (2024). LLMTemporalComparator: A Tool for Analysing Differences in Temporal Adaptations of Large Language Models. *arXiv Computer Science*. https://doi.org/10.48550/arXiv.2410.04195

Haddaway, N. R., Page, M. J., Pritchard, C. C., & McGuinness, L. A. (2022). PRISMA2020: An R package and Shiny app for producing PRISMA 2020-compliant flow diagrams, with interactivity for optimised digital transparency and Open Synthesis. *Campbell Systematic Reviews*, *18*(2), e1230. https://doi.org/10.1002/cl2.1230

Haindl, P., & Weinberger, G. (2024). Students' Experiences of Using ChatGPT in an Undergraduate Programming Course. *IEEE Access*, *12*, 43519-43529. https://doi.org/10.1109/ACCESS.2024.3380909

Harzing, A.W. (2007). *Publish or Perish*. https://harzing.com/resources/publish-or-perish

Jošt, G., Taneski, V., & Karakatič, S. (2024). The Impact of Large Language Models on Programming Education and Student Learning Outcomes. *Applied Sciences*, *14*(10), 4115. https://doi.org/10.3390/app14104115

Kau, A., He, X., Nambissan, A., Astudillo, A., Yin, H., & Aryani, A. (2024). Combining Knowledge Graphs and Large Language Models. *arXiv Computer Science*. https://doi.org/10.48550/arXiv.2407.06564

Kazemitabaar, M., Chow, J., Ma, C. K. T., Ericson, B. J., Weintrop, D., & Grossman, T. (2023). *Studying the effect of AI Code Generators on Supporting Novice Learners in Introductory Programming*. Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, 1-23. https://doi.org/10.1145/3544548.3580919

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J. M., Hróbjartsson, A., Lalu, M. M., Li, T., Loder, E. W., Mayo-Wilson, E., McDonald, S., … Alonso-Fernández, S. (2021). Declaración PRISMA 2020: Una guía actualizada para la publicación de revisiones sistemáticas. *Revista Española de Cardiología*, *74*(9), 790-799. https://doi.org/10.1016/j.recesp.2021.06.016

Pornprasit, C., & Tantithamthavorn, C. (2024). Fine-tuning and prompt engineering for large language models-based code review automation. *Information and Software Technology*, *175*, 107523. https://doi.org/10.1016/j.infsof.2024.107523

Tao, N., Ventresque, A., Nallur, V., & Saber, T. (2024). Enhancing Program Synthesis with Large Language Models Using Many-Objective Grammar-Guided Genetic Programming. *Algorithms*, *17*(7), 287. https://doi.org/10.3390/a17070287

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *arXiv Computer Science.* https://doi.org/10.48550/arXiv.1706.03762

Wang, R., Xu, S., Tian, Y., Ji, X., Sun, X., & Jiang, S. (2024). SCL-CVD: Supervised contrastive learning for code vulnerability detection via GraphCodeBERT. *Computers & Security*, *145*, 103994. https://doi.org/10.1016/j.cose.2024.103994

Weber, I. (2024). Large Language Models as Software Components: A Taxonomy for LLM-Integrated Applications. *arXiv Computer Science*. https://doi.org/10.48550/ARXIV.2406.10300

Yun, S., Lin, S., Gu, X., & Shen, B. (2024). Project-specific code summarization with in-context learning. *Journal of Systems and Software*, *216*, 112149. https://doi.org/10.1016/j.jss.2024.112149

Zhou, X., Liang, P., Zhang, B., Li, Z., Ahmad, A., Shahin, M., & Waseem, M. (2025). Exploring the problems, their causes and solutions of AI pair programming: A study on GitHub and Stack Overflow. *Journal of Systems and Software*, *219*, 112204. https://doi.org/10.1016/j.jss.2024.112204