INVESTICACION APLICADA 1: USOS DE GITHUB Y COPILOT

Bryan Willian Hernández Artiaga HA231511 David Isaac Segura Sánchez SS231442 Omar Arturo Maldonado Guzman MG220764 Yesenia Lizbeth Hernández Portillo HP240153 Cristina Aracely Perez Lué PL240092

|  |  |
| --- | --- |
| ***Resumen*—En este documento hablaremos sobre**  **Copilot es un compañero de inteligencia artificial creado por Microsoft diseñado para ayudarte en una variedad de tareas y brindar información precisa y útil. Puedes pensar en mí como un asistente siempre disponible para responder preguntas, proporcionar ideas, generar contenido creativo, ofrecer soporte y mucho más.** | . |

# I.I. INTRODUCCIÓN

# GitHub Copilot es una herramienta revolucionaria desarrollada conjuntamente por GitHub y OpenAI que utiliza inteligencia artificial para ayudar a los desarrolladores a escribir código de manera más eficiente. Desde su lanzamiento inicial, Copilot ha transformado la forma en que los programadores abordan la escritura de código, ofreciendo sugerencias contextuales y automáticas basadas en modelos avanzados de aprendizaje profundo. Esta exposición abordará la historia y desarrollo de GitHub Copilot, su funcionamiento técnico, su impacto en la productividad del desarrollador, las ventajas y limitaciones de su uso, una comparación con otras herramientas similares, y los requerimientos de hardware y software necesarios para su óptimo rendimiento.

# II. HISTORIA Y DESARROLLO DE GITHUB COPILOT

A. Orígenes y Evolución de GitHub Copilot GitHub Copilot es un asistente de programación basado en inteligencia artificial desarrollado por GitHub en colaboración con OpenAI. Desde su lanzamiento inicial, ha evolucionado para ofrecer sugerencias de código en múltiples lenguajes de programación y ha sido adoptado por desarrolladores de todo el mundo.

B. Colaboración entre GitHub y OpenAI La colaboración entre GitHub y OpenAI ha sido fundamental para el desarrollo de Copilot. OpenAI proporcionó el modelo de lenguaje GPT-3, el cual fue entrenado con una vasta cantidad de código y datos relevantes para mejorar las sugerencias de programación.

C. Versiones y Mejoras desde su Lanzamiento Inicial Desde su lanzamiento, GitHub Copilot ha recibido múltiples actualizaciones y mejoras. Estas incluyen la expansión a nuevos lenguajes de programación, mejoras en la precisión de las sugerencias y la capacidad de entender el contexto del código para ofrecer mejores resultados.

Figura[1]. Github copilot

II. FUNCIONAMIENTO TÉCNICO DE GITHUB COPILOT

A. Arquitectura y Modelo de Inteligencia Artificial Detrás de Copilot Copilot se basa en el modelo de lenguaje GPT-3 de OpenAI. Este modelo utiliza una arquitectura de red neuronal profunda para generar sugerencias de código basadas en el contexto del código proporcionado por el usuario

B. Algoritmos y Técnicas de Aprendizaje Profundo Utilizados El modelo utiliza técnicas avanzadas de aprendizaje profundo, incluyendo el entrenamiento supervisado y no supervisado, para mejorar continuamente sus capacidades. Los algoritmos permiten a Copilot aprender de grandes volúmenes de datos y adaptarse a diferentes estilos de programación.

C. Cómo Copilot se Integra con Visual Studio Code a Nivel Técnico La integración de Copilot con Visual Studio Code se realiza a través de una extensión. Esta extensión permite que Copilot analice el código en tiempo real y ofrezca sugerencias contextuales directamente en el entorno de desarrollo integrado (IDE).

IV. VENTAJAS DE UTILIZAR GITHUB COPILOT

A. Casos de Uso Específicos Donde Copilot ha Demostrado Ser Beneficioso Copilot ha demostrado ser especialmente útil en tareas repetitivas, en la generación de código estándar, y en la asistencia para la resolución de problemas comunes en la programación.

B. Mejora en la Eficiencia de Escritura de Código Al proporcionar sugerencias de código en tiempo real, Copilot reduce el tiempo necesario para escribir código manualmente y mejora la eficiencia del proceso de desarrollo.

C. Reducción del Tiempo en Tareas Repetitivas y Comunes Copilot puede automatizar tareas repetitivas y comunes, permitiendo a los desarrolladores enfocarse en problemas más complejos y creativos.

D. Asistencia en la Resolución de Problemas Complejos y Depuración Copilot ofrece sugerencias para la depuración de código y la resolución de problemas complejos, ayudando a los desarrolladores a encontrar soluciones de manera más rápida y eficiente.

**V. LIMITACIONES Y DESAFÍOS DE GITHUB COPILOT**

A. Dependencia Excesiva y su Impacto en el Desarrollo de Habilidades El uso excesivo de Copilot puede llevar a una dependencia que podría afectar el desarrollo de habilidades fundamentales en programación.

B. Problemas de Seguridad y Privacidad de Datos Existen preocupaciones sobre la seguridad y la privacidad de los datos cuando se utilizan herramientas de inteligencia artificial como Copilot, especialmente en entornos sensibles.

C. Calidad y Relevancia de las Sugerencias Proporcionadas por Copilot Aunque Copilot ofrece sugerencias útiles, no siempre son perfectas. La calidad y relevancia de las sugerencias pueden variar y requieren revisión humana.

D. Reacciones y Críticas de la Comunidad de Desarrolladores La comunidad de desarrolladores ha tenido reacciones mixtas sobre Copilot. Mientras que algunos lo ven como una herramienta revolucionaria, otros plantean preocupaciones sobre la dependencia y la calidad del código generado.

**VI. COMPARACIÓN CON OTRAS HERRAMIENTAS SIMILARES**

A. Evaluación de Otras Herramientas de Asistencia al Desarrollo Basadas en IA Existen varias herramientas de asistencia al desarrollo basadas en IA, como TabNine y Kite, que compiten con Copilot en términos de características y rendimiento.

B. Comparación de Características, Rendimiento y Aceptación en la Comunidad Cada herramienta tiene sus propias ventajas y desventajas. Copilot destaca por su integración con GitHub y Visual Studio Code, mientras que otras herramientas pueden tener mejor rendimiento en diferentes entornos.

C. Ventajas y Desventajas Relativas de Copilot Frente a sus Competidores Copilot ofrece una integración profunda con el ecosistema de GitHub, lo que le da una ventaja en términos de colaboración y gestión de código. Sin embargo, otras herramientas pueden ofrecer mejor rendimiento en situaciones específicas.

**VII. REQUERIMIENTOS DE HARDWARE Y SOFTWARE**

A. Especificaciones Técnicas Necesarias para un Rendimiento Óptimo Para un rendimiento óptimo, se recomienda utilizar Copilot en un equipo con un procesador moderno, suficiente memoria RAM y almacenamiento rápido. Además, es importante contar con una conexión a internet estable.

B. Comparación de Rendimiento en Diferentes Configuraciones de Hardware y Sistemas Operativos El rendimiento de Copilot puede variar según la configuración de hardware y el sistema operativo. Es importante probar la herramienta en el entorno específico de desarrollo para asegurarse de su eficiencia.

C. Consideraciones sobre el Consumo de Recursos y la Eficiencia Copilot puede consumir una cantidad significativa de recursos del sistema, por lo que es crucial monitorizar el rendimiento y ajustar la configuración según sea necesario para mantener la eficiencia.

Espero que esta estructura te sea útil para agregar la información a tu documento IEEE. Si necesitas más ayuda o detalles adicionales, ¡no dudes en pedírmelo!

# IV. CONCLUSIONES

GitHub Copilot ha demostrado ser una herramienta innovadora y valiosa para el desarrollo web, brindando a los desarrolladores la capacidad de escribir código de manera más rápida y eficiente. A través de su inteligencia artificial avanzada y su integración con Visual Studio Code, Copilot ha mejorado la productividad de los desarrolladores al automatizar tareas repetitivas y ofrecer sugerencias contextuales de alta calidad. Sin embargo, es importante reconocer las limitaciones y desafíos asociados con su uso, como la dependencia excesiva las preocupaciones de seguridad. A pesar de estos desafíos, las ventajas de utilizar Copilot superan sus desventajas, y su impacto positivo en la industria del desarrollo de software es innegable.

## RECONOCIMIENTOS

Esta sección sigue el formato regular del resto del documento. La única observación es notar que el título no está numerado.

En esta sección se agregan agradecimientos a personas que colaboraron en el proyecto pero que no figuran como autores del artículo.

## REFERENCIAS

1.  S. M. Metev and V. P. Veiko, *Laser Assisted Microtechnology*, 2nd ed., R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998.
2.  J. Breckling, Ed., *The Analysis of Directional Time Series: Applications to Wind Speed and Direction*, ser. Lecture Notes in Statistics. Berlin, Germany: Springer, 1989, vol. 61.
3.  S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok, "A novel ultrathin elevated channel low-temperature poly-Si TFT," *IEEE Electron Device Lett.*, vol. 20, pp. 569–571, Nov. 1999.
4.  M. Wegmuller, J. P. von der Weid, P. Oberson, and N. Gisin, "High resolution fiber distributed measurements with coherent OFDR," in *Proc. ECOC'00*, 2000, paper 11.3.4, p. 109.
5.  R. E. Sorace, V. S. Reinhardt, and S. A. Vaughn, "High-speed digital-to-RF converter," U.S. Patent 5 668 842, Sept. 16, 1997.
6.  (2002) The IEEE website. [Online]. Available: http://www.ieee.org/
7.  M. Shell. (2002) IEEEtran homepage on CTAN. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/supported/IEEEtran/
8.  FLEXChip Signal Processor (MC68175/D), Motorola, 1996.
9.  “PDCA12-70 data sheet,” Opto Speed SA, Mezzovico, Switzerland.
10.  A. Karnik, "Performance of TCP congestion control with rate feedback: TCP/ABR and rate adaptive TCP/IP," M. Eng. thesis, Indian Institute of Science, Bangalore, India, Jan. 1999.
11.  J. Padhye, V. Firoiu, and D. Towsley, "A stochastic model of TCP Reno congestion avoidance and control," Univ. of Massachusetts, Amherst, MA, CMPSCI Tech. Rep. 99-02, 1999.
12.  *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11, 1997.