UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERÍA

ASIGNATURA:

DISEÑO Y PROGRAMACIÓN DE SOFTWARE MULTIPLATAFORMA

GRUPO:

02T

ACTIVIDAD:

INVESTIGACIÓN APLICADA 1: COPILOT EN VISUAL STUDIO CODE.

DOCENTE:

ING. ALEXANDER ALBERTO SIGUENZA CAMPOS

INTEGRANTES:

NOMBRE	**CARNÉ**
ALEJANDRA DEL CARMEN PONCE LÓPEZ	PL210665
DAVID ELEAZAR GUZMÁN GUARDADO	GG171680
MARÍA JOSÉ TORRES MOLINA	TM210066
LEÓN MARROQUÍN, RONALD VLADIMIR	LM192213
OSWALDO ENRIQUE HERNÁNDEZ TORRES	HT192210
CARLOS ALBERTO AMAYA CÁRCAMO	AC220139

1. Historia y Desarrollo de GitHub Copilot

a. Orígenes y evolución de GitHub Copilot.

GitHub Copilot es una herramienta de inteligencia artificial desarrollada en colaboración entre GitHub y OpenAI, diseñada para asistir a los programadores en la escritura de código. Lanzada en vista previa técnica en junio de 2021, Copilot representa un avance significativo en el uso de la inteligencia artificial para el desarrollo de software. Su evolución y creación se basan en años de avances en el aprendizaje profundo y el procesamiento del lenguaje natural.

La idea detrás de GitHub Copilot surgió de la necesidad de hacer el proceso de codificación más eficiente y accesible. Tradicionalmente, escribir código puede ser un proceso arduo y repetitivo, con desarrolladores a menudo teniendo que buscar soluciones comunes y patrones de diseño. La visión de GitHub y OpenAI fue crear una herramienta que pudiera entender el contexto del código en el que trabaja un desarrollador y sugerir fragmentos de código, completando funciones o incluso escribiendo algoritmos completos basados en descripciones en lenguaje natural.

Para lograr esto, Copilot utiliza Codex, un modelo de inteligencia artificial basado en GPT-3 de OpenAI. GPT-3 es uno de los modelos de lenguaje más avanzados, entrenado en una amplia variedad de textos de Internet, lo que le permite comprender y generar texto con una sorprendente precisión. Codex, una variante de GPT-3, fue específicamente afinado para la programación, entrenado con una cantidad masiva de código fuente disponible públicamente, incluidos los repositorios de código abierto en GitHub.

b. Colaboración entre GitHub y OpenAI.

El desarrollo de Copilot involucró múltiples desafíos, tanto técnicos como éticos. Desde el punto de vista técnico, entrenar un modelo lo suficientemente potente como para entender y generar código relevante y eficiente requirió un manejo cuidadoso de los datos y una gran cantidad de poder de cómputo. El equipo de OpenAI trabajó para asegurar que Codex no solo generara código que funcionara, sino que también siguiera las mejores prácticas de codificación, algo que se ha ido mejorando continuamente con el feedback de los usuarios durante la vista previa técnica.

Desde una perspectiva ética, el uso de código fuente público para entrenar a Copilot planteó preguntas sobre derechos de autor y licencias. GitHub y OpenAI han trabajado para abordar estas preocupaciones, asegurando que Copilot no replique directamente grandes fragmentos de código de sus fuentes de entrenamiento y alentando a los desarrolladores a revisar y comprender el código sugerido antes de integrarlo en sus proyectos.

En cuanto a su funcionalidad, Copilot se integra directamente en los entornos de desarrollo integrados (IDEs) más populares, como Visual Studio Code, ofreciendo sugerencias de código en tiempo real. A medida que el desarrollador escribe, Copilot analiza el contexto y sugiere posibles continuaciones, lo que puede incluir desde líneas individuales de código hasta funciones completas. Esta capacidad de autocompletar y generar código no solo ahorra tiempo, sino que también ayuda a los desarrolladores a aprender nuevas técnicas y patrones de diseño.

c. Versiones y mejoras desde su lanzamiento inicial.

El lanzamiento de Copilot fue recibido con entusiasmo y también con cierta cautela dentro de la comunidad de desarrolladores. Mientras muchos aprecian la ayuda y la eficiencia que ofrece, otros han expresado preocupaciones sobre la dependencia de la inteligencia artificial y la potencial deshumanización del proceso de desarrollo de software. Sin embargo, a medida que la herramienta continúa evolucionando y mejorando, parece claro que GitHub Copilot es un vistazo al futuro de la programación asistida por inteligencia artificial.

2. Funcionamiento Técnico de GitHub Copilot

GitHub Copilot funciona mediante el uso de modelos avanzados de aprendizaje automático entrenados en un amplio conjunto de datos de código disponible a nivel público de repositorios de GitHub. A medida que escribe código, la inteligencia artificial analiza el contexto y proporciona sugerencias relevantes en tiempo real. También puede recibir sugerencias escribiendo un comentario de lenguaje natural que describa lo que desea que haga el código.

También puede usar GitHub Copilot para convertir comentarios en código, crear pruebas unitarias, crear consultas SQL, etc.

a. Arquitectura y modelo de inteligencia artificial detrás de Copilot.

GitHub Copilot, lanzado en junio de 2021, es una herramienta de autocompletado de código basada en inteligencia artificial. Su arquitectura y modelo se fundamentan en algunos componentes clave:

- Modelo de Lenguaje Principal: GPT-3

  GitHub Copilot utiliza el modelo de lenguaje GPT-3 (Generative Pre-trained Transformer 3) desarrollado por OpenAI. GPT-3 es una de las versiones más avanzadas de la serie de modelos de transformers, caracterizado por:

  Tamaño: GPT-3 tiene 175 mil millones de parámetros, lo que le permite entender y generar texto con alta fluidez y coherencia.

Entrenamiento: Se entrenó en una amplia gama de textos disponibles públicamente en internet, lo que le permite tener una comprensión generalizada de diversos temas y estilos de escritura.

- Codex

  Copilot está específicamente impulsado por una variante del modelo GPT-3 llamada Codex. Codex está afinado y especializado para entender y generar código fuente. Se diferencia en algunos aspectos clave:

  Especialización en código: Codex está entrenado con una gran cantidad de código fuente de diferentes lenguajes de programación, lo que mejora su capacidad para comprender y generar fragmentos de código.

  Desempeño en tareas de programación: Codex es particularmente bueno en la generación de código, la comprensión de consultas de código y el autocompletado en contextos de programación.

b. Algoritmos y técnicas de aprendizaje profundo utilizados.

GitHub Copilot, impulsado por el modelo Codex de OpenAI, utiliza varias técnicas avanzadas de aprendizaje profundo y algoritmos para proporcionar autocompletado y generación de código.

- Modelos de transformers

  Arquitectura de Transformer: La base de Copilot y Codex es la arquitectura de transformers, presentada en el artículo "Attention Is All You Need" por Vaswani et al. (2017). Esta arquitectura utiliza mecanismos de atención para procesar secuencias de datos, permitiendo que el modelo capture relaciones de largo alcance en el texto.

- Modelos de lenguaje preentrenados

  GPT (Generative Pre-trained Transformer): Codex está basado en GPT-3, una versión avanzada de la arquitectura transformer. GPT-3 es un modelo de lenguaje preentrenado con 175 mil millones de parámetros. El preentrenamiento en grandes corpus de texto permite al modelo aprender representaciones ricas del lenguaje y sus estructuras.

  Codex: Es una variante de GPT-3 adaptada específicamente para el entendimiento y generación de código. Aunque utiliza la misma arquitectura básica, Codex ha sido entrenado con un gran corpus de código fuente, lo que lo hace más competente en tareas relacionadas con la programación.

- Entrenamiento y ajuste fino

  Preentrenamiento: GPT-3 y Codex se entrenan inicialmente en grandes volúmenes de datos textuales y código fuente. El preentrenamiento permite que el modelo aprenda patrones lingüísticos y de programación de manera general.

  Fine-Tuning (Ajuste Fino): Codex pasa por una fase de ajuste fino adicional utilizando conjuntos de datos de código más específicos. Este proceso adapta el modelo para generar y entender mejor el código en varios lenguajes de programación.

- Mecanismo de atención

  Self-Attention: El mecanismo de atención en transformers permite al modelo ponderar diferentes partes de la entrada para capturar dependencias contextuales. Esto es esencial para entender el contexto en el código fuente y generar sugerencias relevantes.

  Multi-Head Attention: Esta técnica permite que el modelo preste atención a diferentes partes del texto de manera simultánea, mejorando la capacidad para manejar múltiples aspectos del contexto del código.

- Tokenización y representación de datos

  Tokenización: Los modelos transformers, incluido Codex, utilizan técnicas de tokenización para convertir texto en secuencias de tokens. Esto puede incluir tokenización subpalabra, que divide palabras en unidades más pequeñas para manejar mejor las variaciones en el código y el lenguaje.

  Embeddings: Los tokens se representan mediante embeddings, que son vectores que capturan la información semántica de las palabras o fragmentos de código. Estos embeddings se usan como entrada para las capas del transformer.

- Generación y decodificación

  Decodificación Autorregresiva: GPT-3 y Codex utilizan un enfoque autorregresivo para la generación de texto. El modelo genera un token a la vez, y cada token generado se utiliza para predecir el siguiente en la secuencia. Esto permite generar código y texto de manera coherente y fluida.

Sampling y Beam Search: Durante la generación, se pueden usar técnicas como el sampling para introducir variabilidad en las sugerencias, o beam search para encontrar las secuencias de tokens más probables basadas en la probabilidad de las predicciones.

- Optimización y regularización

  Optimización: Algoritmos de optimización como Adam se utilizan para ajustar los parámetros del modelo durante el entrenamiento. Estos algoritmos ayudan a minimizar la función de pérdida y mejorar la precisión del modelo.

  Regularización: Técnicas como dropout y normalización de capas se aplican para evitar el sobreajuste y mejorar la capacidad del modelo para generalizar a datos no vistos.

c. Cómo copilot se integra con visual studio code a nivel técnico.

GitHub Copilot se integra con Visual Studio Code (VS Code) mediante una serie de técnicas y mecanismos técnicos. A nivel técnico, la integración de Copilot con VS Code involucra varios componentes clave:

- Extensión de VS Code

  Extensión Oficial: GitHub Copilot se integra en VS Code a través de una extensión específica. Esta extensión se instala desde el Marketplace de VS Code y actúa como un puente entre el editor de código y el servicio de Copilot en la nube.

  Instalación y Configuración: Puedes instalar la extensión de Copilot desde el Visual Studio Code Marketplace.

  Después de la instalación, la extensión se configura automáticamente para comenzar a ofrecer sugerencias y autocompletado basado en el contexto del código que estás escribiendo.

- Comunicación con el Servidor de Copilot

  API de Comunicación: La extensión de Copilot se comunica con los servidores de GitHub Copilot a través de una API que envía fragmentos de código y recibe sugerencias. Este proceso implica:

  Solicitud de Sugerencias: La extensión envía el contexto actual del código (por ejemplo, las líneas de código que se están escribiendo y el contexto circundante) al servidor de Copilot.

Recepción de Respuestas: El servidor de Copilot procesa el contexto usando el modelo Codex y devuelve sugerencias de código que se muestran en el editor.

Autenticación y Seguridad: La comunicación entre la extensión y los servidores se realiza a través de conexiones seguras (HTTPS) para garantizar la privacidad y la seguridad del código del usuario.

- Generación y presentación de sugerencias

  Generación de Código: Una vez que el modelo Codex en la nube recibe el contexto del código, genera sugerencias en función de su entrenamiento previo y el contexto proporcionado. Las sugerencias pueden incluir:

  Autocompletado de Fragmentos: Sugerencias para completar líneas o bloques de código.

  Códigos Enteros: Propuestas para funciones o estructuras completas basadas en el patrón observado.

  Incorporación en el Editor: La extensión de Copilot recibe las sugerencias generadas y las presenta en el editor de VS Code. La interfaz de usuario de la extensión permite al usuario:

  Aceptar Sugerencias: Insertar el código sugerido en el archivo en el que están trabajando.

  Rechazar o Ignorar: Descartar las sugerencias si no son útiles.

- Contexto y análisis

  Contexto del Código: La extensión de Copilot envía información relevante del contexto del código, como:

  Contenido del Archivo: Parte del contenido del archivo en el que el usuario está trabajando.

  Estructura del Código: Información sobre la estructura y el flujo del código para proporcionar sugerencias más precisas.

  Análisis y Optimización: Copilot utiliza técnicas de procesamiento del lenguaje natural (NLP) y aprendizaje profundo para interpretar el contexto del código y generar sugerencias útiles. La calidad y relevancia de las sugerencias se basan en el modelo Codex y el entrenamiento previo.

- Experiencia de usuario y configuración

  Interfaz de Usuario: La extensión de Copilot está diseñada para integrarse de manera fluida con la interfaz de VS Code. Ofrece una experiencia de usuario que incluye:

  Ventanas de Sugerencias: Cuadros emergentes que muestran las sugerencias de código.

  Comandos y Atajos: Comandos y atajos de teclado para aceptar o rechazar sugerencias rápidamente.

  Configuración Personalizada: Los usuarios pueden configurar ciertos aspectos de la extensión, como habilitar o deshabilitar Copilot para archivos específicos, ajustar el nivel de sugestión, y personalizar cómo se presentan las sugerencias.

- Mecanismos de feedback y aprendizaje

  Feedback del Usuario: Los usuarios pueden proporcionar feedback sobre la calidad de las sugerencias, lo que puede ayudar a mejorar el modelo y el servicio en general.

  Actualizaciones y Mejoras: GitHub puede actualizar la extensión y el modelo basado en el feedback recibido y en avances en la tecnología de modelos de lenguaje.

3. Impacto en la productividad del desarrollador

a. Estudios de caso y análisis comparativos pre y post-integración de Copilot

Los datos del estudio de Microsoft revelan que los beneficios de productividad son tangibles. Se utilizó una combinación de encuestas y experimentos para entender cómo Copilot está transformando el trabajo.

- **Productividad mejorada:** El 70% de los usuarios de Copilot afirmaron que eran más productivos, y el 68% dijeron que la calidad de su trabajo mejoró.

- **Ahorro de tiempo:** Los usuarios completaron tareas un 29% más rápido en general. En particular, pudieron ponerse al día con una reunión perdida casi 4 veces más rápido.

- **Eficiencia en el correo electrónico:** El 64% de los usuarios dijeron que Copilot les ayudó a pasar menos tiempo procesando correos electrónicos.

- **Redacción eficiente:** El 85% de los usuarios dijeron que Copilot les ayudó a llegar a un buen primer borrador más rápidamente.

- **Acceso a información:** El 75% de los usuarios dijeron que Copilot les ahorró tiempo al encontrar lo que necesitaban en sus archivos.

– Casos de uso

Microsoft también exploró el impacto de Copilot en funciones específicas como ventas y servicio al cliente, obteniendo resultados muy positivos:

- **Ventas**

En una encuesta realizada a 133 vendedores de Microsoft, se descubrió que Copilot para Ventas les ahorraba en promedio 90 minutos a la semana. Además, el 83% dijo que Copilot les hacía más productivos, y el 67% afirmó que podían pasar más tiempo con sus clientes gracias a la herramienta.

- **Servicio al cliente**

En el ámbito del servicio al cliente, se encontró que los agentes que utilizaban Copilot podían resolver casos un 12% más rápido. Este ahorro de tiempo no solo mejora la eficiencia del servicio al cliente, sino que también contribuye a una mayor satisfacción del cliente.

– Análisis comparativos pre y post-integración de Copilot

- **Antes de la integración de Copilot**

  - **Funcionalidad de autocompletado:**

    ✓ Básico: La funcionalidad de autocompletado en VS Code antes de Copilot era básica, basada en el análisis sintáctico y algunas inferencias contextuales.

    ✓ Dependencia de snippets: Los desarrolladores dependían mucho de los snippets predefinidos y de su propia memoria para completar el código.

- **Productividad:**

  - ✓ Moderada: Los desarrolladores podían aprovechar herramientas como linters y snippets, pero la generación de código aún era manual y a menudo lenta.

  - ✓ Curva de aprendizaje: Los nuevos desarrolladores enfrentaban una curva de aprendizaje más pronunciada debido a la necesidad de memorizar sintaxis y patrones comunes.

- **Colaboración:**

  - ✓ Manual: La colaboración se realizaba mediante revisiones de código y comentarios en los sistemas de control de versiones.

  - ✓ Poca asistencia automatizada: No había herramientas significativas que asistieran automáticamente en la colaboración en tiempo real.

- **Innovación:**

  - ✓ Limitada: Las innovaciones en las herramientas de desarrollo se centraban en mejoras incrementales de funcionalidades existentes como el debugging, el manejo de proyectos, y la integración con sistemas de control de versiones.

- **Después de la integración de Copilot**

- **Funcionalidad de autocompletado:**

  - ✓ **Avanzado:** Copilot ofrece autocompletado basado en IA, capaz de entender el contexto del código y proporcionar sugerencias inteligentes y completas.

  - ✓ **Generación de código:** Copilot puede generar bloques enteros de código basándose en comentarios y fragmentos de código anteriores, lo que agiliza significativamente el desarrollo.

- **Productividad:**

  - ✓ **Alta:** La generación automática de código y las sugerencias contextuales permiten a los desarrolladores trabajar más rápido y reducir errores.

- ✓ **Reducción de la curva de aprendizaje:** Los nuevos desarrolladores pueden comenzar a escribir código eficaz más rápidamente gracias a las sugerencias contextuales y ejemplos proporcionados por Copilot.

- **Colaboración:**

  - ✓ **Asistencia automatizada:** Copilot facilita la colaboración en tiempo real al proporcionar sugerencias de código y soluciones a problemas comunes, lo que puede ser útil durante las sesiones de programación en pareja.

  - ✓ **Revisiones de código:** La calidad de las revisiones de código puede mejorar, ya que Copilot ayuda a mantener estándares de codificación y detectar posibles errores o mejoras.

- **Innovación:**

  - ✓ **Inteligencia artificial:** La integración de IA en el desarrollo de software abre nuevas posibilidades para herramientas más inteligentes y personalizadas.

  - ✓ **Ecosistema en crecimiento:** La comunidad de desarrollo puede beneficiarse de las continuas mejoras y actualizaciones de Copilot, así como de nuevas herramientas basadas en IA que se integran en VS Code.

b. Métodos de medición de productividad en el desarrollo de software.

- **Líneas de código (LOC) generadas por Copilot**

  Mide el número de líneas de código generadas automáticamente por Copilot.

- **Function points**

  Evalúa la cantidad y complejidad de las funciones implementadas con la ayuda de Copilot.

- **Velocidad en scrum**

  Calcula la cantidad de puntos de historia completados en un sprint con la asistencia de Copilot.

- **Defect cycle time**

  Mide el tiempo desde que se reporta un defecto hasta que se cierra, con énfasis en cómo Copilot ayudó a resolver los defectos.

- **Feedback de desarrolladores**

  Recoge opiniones de los desarrolladores sobre la utilidad y eficiencia de Copilot en sus tareas diarias.

c. Percepciones y experiencias de desarrolladores que utilizan Copilot.

Juan Font

Juan Font, desarrollador para la Agencia Espacial Europea, ve mucho potencial en el servicio. Comenta que le ha sorprendido cómo en bloques condicionales (if/else), Copilot es capaz de predecir el mensaje de registro que quiere añadir basándose a las condiciones del bloque.

También destaca cómo la herramienta sugiere un nombre adecuado para una función sin que no se haya escrito ni una línea de código de ella, solamente a partir de un comentario que escribas. Además, ha encontrado mucha utilidad a la hora de escribir código repetitivo (boilerplate en la jerga de programadores). De todos modos, Juan es claro y califica Copilot de "espectacular", con la impresión de que se va a convertir en algo tan básico y cotidiano como el completado de código automático que todos los entornos de desarrollo integran para evitar errores tipográficos en el código.

Pablo Serrano

Pablo Serrano, ingeniero especializado en sistemas. Para él la herramienta no va a reemplazar a los programadores ni mucho menos, pero sí que va a quitarles mucho trabajo y muchas búsquedas en Stackoverflow, un portal online en el que muchos programadores resuelven dudas. Copilot "te apabulla" con sugerencias de código, sigue Pablo, considerando que "va a quitar bastantes horas de curro" a los programadores especialmente si su código es sencillo.

Bruno, un desarrollador especialista en React que actualmente trabaja en un bot de Discord, también nos ha contado brevemente su experiencia después de dos días probando Github Copilot. Su sensación es "muy buena" con las pruebas que ha hecho centrándose en React y Node, y afirma que la herramienta le ha facilitado mucho su trabajo.

4. Ventajas de utilizar GitHub Copilot

GitHub Copilot ofrece varias ventajas para los desarrolladores de software. Aquí te detallo cada uno de los puntos mencionados:

a. Casos de uso específicos donde Copilot ha demostrado ser beneficioso:

- Generación de código boilerplate: Copilot es útil para generar automáticamente código repetitivo y boilerplate, lo que ahorra tiempo en la escritura de código estándar.
- Soporte para nuevos lenguajes y tecnologías: Para desarrolladores que están aprendiendo nuevos lenguajes o frameworks, Copilot puede proporcionar ejemplos y plantillas relevantes.

b. Mejora en la eficiencia de escritura de código:

- Sugerencias en tiempo real: Copilot ofrece sugerencias de código en tiempo real mientras el desarrollador escribe, lo que puede acelerar el proceso de codificación y reducir la necesidad de buscar en documentación externa.
- Completar funciones y métodos: Puede ayudar a completar funciones o métodos basados en el contexto del código actual, lo que agiliza la programación.

c. Reducción del tiempo en tareas repetitivas y comunes:

- Autocompletado inteligente: Ayuda a reducir el tiempo dedicado a escribir código repetitivo o común al proporcionar autocompletado de fragmentos de código que son frecuentemente utilizados.
- Generación de comentarios y documentación: Copilot puede sugerir comentarios y documentación relevantes para el código, facilitando la comprensión y el mantenimiento del mismo con la referencias necesarias.

d. Asistencia en la resolución de problemas complejos y depuración:

- Sugerencias para resolución de errores: Puede ofrecer sugerencias para solucionar errores comunes o problemas de lógica en el código, ayudando a depurar más rápidamente.
- Ejemplos de código y patrones: Proporciona ejemplos y patrones que pueden ser útiles para entender cómo abordar problemas complejos o implementar soluciones.

Estas ventajas demuestran cómo GitHub Copilot puede ser una herramienta valiosa para mejorar la productividad y la eficiencia en el desarrollo de software.

5. Limitaciones y desafíos de GitHub Copilot

Si bien es una herramienta revolucionaria, no está exenta de limitaciones y desafíos que debemos considerar. Puede provocar errores o soluciones poco fiables para los desarrolladores ya que deben probar y verificar el código generado para evitar problemas. Es una herramienta en manos de los desarrolladores, no un reemplazo.

a. Dependencia excesiva y su impacto en el desarrollo de habilidades

- Reducción en la comprensión profunda

- Aceptación automática: Al recibir sugerencias de código de Copilot, los desarrolladores pueden sentirse tentados a aceptar automáticamente lo que se les propone sin realmente entender el por qué o cómo funciona el código. Esto puede llevar a una comprensión superficial de los conceptos y estructuras subyacentes, impidiendo un aprendizaje profundo y duradero.

- Menos exploración: Los desarrolladores podrían explorar menos las diferentes maneras de resolver un problema porque Copilot les da una solución rápidamente. Esto limita la práctica de investigar, probar, y aprender nuevas técnicas o enfoques, que es esencial para desarrollar una comprensión integral de la programación.

- Disminución en la capacidad de resolución de problemas

- Dependencia en la herramienta: Si los desarrolladores confían en que Copilot les proporcionará la solución correcta, podrían evitar enfrentar desafíos complejos por sí mismos. La capacidad para resolver problemas es una de las habilidades más valiosas en la programación, y se desarrolla enfrentando y superando desafíos. Provocando que sean menos competentes en resolver problemas.

- Menos práctica en la codificación desde cero: La programación efectiva requiere la capacidad de escribir código desde cero. Con Copilot los desarrolladores podrían perder la práctica de crear soluciones propias, lo que es crucial para mejorar la habilidad de codificación.

- Impacto en la creatividad y la innovación

- Soluciones predefinidas: GitHub Copilot tiende a sugerir soluciones basadas en patrones comunes y código existente en repositorios públicos. Esto podría llevar a que los desarrolladores dependan de estos patrones predefinidos, en lugar de explorar y crear nuevas soluciones innovadoras. La creatividad en la programación es clave para desarrollar software eficiente, seguro y con un alto rendimiento, y la dependencia de una herramienta puede limitar la capacidad de innovar.

b. Problemas de seguridad y privacidad de datos

Los problemas de seguridad y privacidad de datos son aspectos críticos que se deben considerar al utilizar GitHub Copilot.

- Exposición involuntaria de código sensible

- Sugerencias de código privado: GitHub Copilot se entrena con una amplia gama de código disponible públicamente en GitHub, lo que incluye tanto código con licencias permisivas como código que podría contener información sensible. Existe el riesgo de que Copilot sugiera fragmentos de código que originalmente provienen de repositorios privados o que contienen datos sensibles, como claves API, contraseñas o información personal, si estos se han subido a repositorios públicos sin las protecciones adecuadas.

- Licencias y propiedad intelectual

- Infracción de derechos de autor: Copilot puede generar código que está basado en fragmentos de código de fuentes que tienen licencias específicas, como GPL o MIT. Si un desarrollador incluye este código sin considerar la licencia original, podría estar infringiendo los derechos de autor o violando los términos de la licencia. Esto plantea un riesgo legal significativo, especialmente en entornos corporativos.

- Ambigüedad en la atribución: Debido a que Copilot no siempre proporciona un contexto claro de dónde proviene el código sugerido, puede ser difícil para los desarrolladores atribuir correctamente el código a su fuente original, lo que puede violar las licencias que requieren atribución.

c. Calidad y relevancia de las sugerencias proporcionadas por Copilot

La calidad y relevancia de las sugerencias proporcionadas por GitHub Copilot son aspectos clave para determinar su efectividad y utilidad en el desarrollo de software.

- Calidad de las sugerencias

- Precisión y efectividad: La calidad de estas sugerencias puede variar considerablemente en algunos casos, Copilot puede proporcionar soluciones que son efectivas y bien estructuradas, pero, en otros puede generar código que es subóptimo, incorrecto o que no se ajusta a las mejores prácticas de programación.

- Consistencia: Puede que funcione bien en ciertos lenguajes o contextos, pero tenga dificultades en otros. Esta inconsistencia puede obligar a los desarrolladores a revisar y corregir las sugerencias, lo que puede reducir el ahorro de tiempo que la herramienta pretende ofrecer.

- Relevancia de las sugerencias

- Contexto del código: Copilot intenta comprender el contexto del código, pero no siempre tiene éxito en capturar la intención del desarrollador o en comprender el propósito específico de un fragmento de código. Esto puede llevar a sugerencias que, aunque técnicamente correctas, no sean relevantes o útiles para el problema.

- Adaptación a necesidades específicas: En muchos casos, las soluciones necesitan ser adaptadas a las necesidades específicas de un proyecto o a las convenciones particulares de un equipo. Esto puede requerir ajustes manuales significativos para que el código generado sea verdaderamente relevante y útil.

– Problemas de redundancia y complejidad

- Código Redundante: A veces, Copilot puede generar código que es innecesariamente largo o que incluye redundancias. Esto puede ocurrir cuando la herramienta intenta cubrir todos los casos posibles o incluye más lógica de la necesaria, lo que puede complicar la base de código y dificultar su mantenimiento.

d. Reacciones y críticas de la comunidad de desarrolladores

Las reacciones y críticas de la comunidad de desarrolladores hacia GitHub Copilot han sido variadas, reflejando tanto entusiasmo como preocupaciones significativas.

– Entusiasmo por la Innovación

- Aumento de la productividad: La herramienta puede acelerar el proceso de escritura de código, sugerir soluciones a problemas comunes y ayudar a automatizar tareas repetitivas, lo que ha sido bien recibido por quienes buscan optimizar su flujo de trabajo

- Facilitación del aprendizaje: Los novatos, ven a Copilot como una valiosa herramienta educativa. Les permite aprender nuevas técnicas y explorar diferentes enfoques para resolver problemas de programación, lo que puede ser un complemento útil en su proceso de aprendizaje.

– Preocupaciones por la calidad y la dependencia

- Calidad variable del código: Copilot no siempre es consistente. Algunos desarrolladores han reportado que, en ocasiones, la herramienta genera código que no sigue las mejores prácticas o que incluye errores, lo que puede ser contraproducente si no se revisa cuidadosamente.

- Dependencia y pérdida de habilidades: Esta dependencia podría afectar su capacidad para resolver problemas por sí mismos y limitar el desarrollo de habilidades críticas, como la capacidad de escribir código desde cero y comprender profundamente los conceptos subyacentes.

– Cuestiones Éticas y Legales

- Problemas de licencias: Muchos desarrolladores han planteado preocupaciones legales de generar código basado en fragmentos de código protegido por derechos de autor. Este problema ha llevado a discusiones sobre la ética de utilizar código de repositorios públicos sin un consentimiento explícito de los autores originales.

- Propiedad Intelectual: Algunos desarrolladores se preguntan si el código generado por la herramienta podría considerarse una infracción de propiedad intelectual, lo que podría tener implicaciones legales para quienes lo utilicen en proyectos comerciales.

6. Comparación con Otras Herramientas Similares

a. Evaluación de otras herramientas de asistencia al desarrollo basadas en IA.

- GitHub Copilot (Base de comparación)

- Popularidad: GitHub Copilot, desarrollado por GitHub en colaboración con OpenAI, ha ganado una gran popularidad debido a la reputación y la base de usuarios de GitHub.

- Ventajas:
  - Integración Directa: Se integra perfectamente con Visual Studio Code, uno de los editores de código más utilizados.
  - Capacidades Avanzadas de IA: Utiliza el modelo de lenguaje GPT-3 de OpenAI, conocido por sus avanzadas capacidades de generación de texto.
  - Apoyo de GitHub: La integración directa con GitHub facilita el acceso a muchos repositorios y ejemplos de código.

- Casos de uso: Autocompletado de código, sugerencias contextuales, generación de funciones y snippets.

- Tabnine

- Popularidad: Tabnine comparte con Github Copilot el liderazgo de herramientas que ofrecen un autocompletado de código basado en IA, lo que le ha permitido establecerse bien en la comunidad de desarrolladores.

- Ventajas:
    - Soporte Multilenguaje: Admite una amplia gama de lenguajes de programación y entornos de desarrollo.
    - Configurabilidad: Los desarrolladores pueden ajustar la herramienta para que se adapte a sus necesidades específicas.
    - Compatibilidad: Compatible con muchos IDEs y editores de código, lo que facilita su adopción.

- Casos de uso: Sugerencias de código, autocompletado, aceleración de la escritura de código.

- Microsoft IntelliCode

- Popularidad: Microsoft IntelliCode es muy popular debido a la gran base de usuarios de Visual Studio y Visual Studio Code.

- Ventajas:
    - Integración Nativa: Al ser una extensión oficial de Microsoft, se integra perfectamente con Visual Studio y Visual Studio Code, dos de los entornos de desarrollo más utilizados en la industria.

    - Sugerencias Basadas en IA: Utiliza modelos de aprendizaje automático para proporcionar sugerencias inteligentes y autocompletado de código.

    - Recomendaciones de Mejores Prácticas: Además de autocompletar el código, IntelliCode también recomienda prácticas de codificación basadas en los patrones de código más comunes en repositorios de código abierto de alta calidad.

    - Soporte para Múltiples Lenguajes: Admite varios lenguajes de programación, incluidos C#, C++, JavaScript, TypeScript y Python.

- Casos de uso: Autocompletado de código, recomendaciones de mejores prácticas, aceleración del desarrollo, soporte para múltiples lenguajes.

b. Comparación de características, rendimiento y aceptación en la comunidad

- GitHub Copilot

- Características:
  - Autocompletado Avanzado: Basado en GPT-3 de OpenAI, proporciona sugerencias avanzadas y completas para el código.
  - Generación de Código: Puede generar funciones completas y bloques de código a partir de comentarios y descripciones.
  - Compatibilidad: Funciona con Visual Studio Code y GitHub Codespaces.
  - Multilenguaje: Soporta varios lenguajes de programación, aunque tiene un rendimiento óptimo en lenguajes populares como Python, JavaScript, TypeScript, Ruby, y Go.
- Rendimiento:
  - Precisión: Alta precisión en la generación de código debido a su modelo basado en GPT-3.
  - Contexto: Excelente capacidad para entender el contexto y generar código relevante.
  - Velocidad: Rápido en la generación de sugerencias y código completo.
- Aceptación en la Comunidad:
  - Popularidad: Alta, especialmente entre los usuarios de GitHub y Visual Studio Code.
  - Comentarios: Generalmente positivos, valorando su capacidad para aumentar la productividad y facilitar la escritura de código.

- Tabnine

- Características:
  - Autocompletado de Código: Utiliza modelos de IA para sugerir fragmentos de código y autocompletar.
  - Soporte Multilenguaje: Admite más de 20 lenguajes de programación.

    - Integración con IDEs: Compatible con múltiples editores y entornos de desarrollo como VS Code, IntelliJ, PyCharm, y más.

    - Configuración Personalizable: Los usuarios pueden ajustar el comportamiento de las sugerencias según sus necesidades.

- Rendimiento:
    - Precisión: Buena precisión en la mayoría de los lenguajes soportados.

    - Contexto: Sugerencias contextuales basadas en el código circundante.

    - Velocidad: Rápido en la generación de sugerencias, aunque puede variar dependiendo del IDE y la configuración.

- Aceptación en la Comunidad:
    - Popularidad: Amplia base de usuarios, especialmente en la comunidad de desarrolladores que utilizan múltiples IDEs.

    - Comentarios: Positivos, apreciando su flexibilidad y soporte para una amplia gama de lenguajes y entornos.

- Microsoft IntelliCode

- Características:
    - Sugerencias Inteligentes: Utiliza modelos de aprendizaje automático para proporcionar sugerencias basadas en las mejores prácticas y patrones de código.

    - Recomendaciones Contextuales: Ofrece recomendaciones contextuales basadas en el código escrito.

    - Compatibilidad con Microsoft: Integración perfecta con Visual Studio y Visual Studio Code.

    - Soporte para Múltiples Lenguajes: Admite lenguajes como C#, C++, JavaScript, TypeScript, y Python.

- Rendimiento:
    - Precisión: Alta precisión, especialmente en lenguajes soportados y cuando se utilizan dentro del ecosistema de Microsoft.

- Contexto: Buen entendimiento del contexto del código, proporcionando sugerencias relevantes.

- Velocidad: Rápido y eficiente, con un impacto mínimo en el rendimiento del IDE.

- Aceptación en la Comunidad:
  - Popularidad: Muy popular entre los usuarios de Visual Studio y Visual Studio Code.

  - Comentarios: Positivos, especialmente por su integración nativa y recomendaciones basadas en mejores prácticas.

.

c. Ventajas y desventajas

Herramienta	Ventajas	Desventajas
GitHub Copilot	- Modelo avanzado de IA. - Generación de código completo. - Integración con GitHub. - Soporte multilenguaje.	- Costo. - Dependencia de Internet, Preocupaciones de privacidad del código.
Tabnine	- Soporte multilenguaje. Compatibilidad con múltiples IDEs. - Configuración personalizable. - Opciones locales.	- Precisión variable. - Costo - Menor contexto global.
Microsoft IntelliCode	- Recomendaciones basadas en mejores prácticas. - Integración nativa con Microsoft. - Alta precisión. - Soporte para varios lenguajes.	- Limitado al ecosistema de Microsoft. - Menos innovador en generación de código completo. - Dependencia de Internet para características avanzadas.

7. Requerimientos de hardware y software

a. Especificaciones técnicas necesarias para un rendimiento óptimo

- Conexión a internet
  - **Requerimiento:** Conexión a Internet estable.
  - **Razón:** GitHub Copilot requiere una conexión constante para enviar y recibir datos desde los servidores de OpenAI.
- Editor de código compatible
  - **Requerimiento:** Un editor de código compatible con GitHub Copilot.
  - **Ejemplos:**
    - **Visual Studio Code (VS Code):** La extensión de GitHub Copilot está especialmente diseñada para VS Code.
    - **Visual Studio:** También soporta GitHub Copilot, especialmente en versiones recientes.
  - **Versión Recomendada:** Mantén el editor actualizado para garantizar compatibilidad y acceso a las últimas características.
- Requisitos del sistema para el editor
  - **Visual Studio Code (VS Code):**
    - **Windows:** Windows 7 o superior.
    - **macOS:** macOS 10.11 (El Capitan) o superior.
    - **Linux:** Dependiendo de la distribución, se recomienda una versión relativamente moderna.
  - **Visual Studio:**
    - **Windows:** Windows 10 o superior, y las versiones de Visual Studio deben ser relativamente recientes para asegurar soporte completo.
- Extensiones y plugins
  - **GitHub Copilot Extension:** Debe instalarse y configurarse en el editor.
  - **Configuración:** Asegúrate de que la extensión esté correctamente configurada y autenticada con tu cuenta de GitHub.

- Requisitos de hardware
    - **CPU:** Cualquier procesador moderno debería ser suficiente, aunque los procesadores más rápidos proporcionarán un rendimiento general más ágil.
    - **RAM:** Se recomienda tener al menos 4 GB de RAM. Sin embargo, 8 GB o más es ideal, especialmente si trabajas en proyectos grandes o utilizas múltiples extensiones.
    - **Espacio en Disco:** Asegúrate de tener suficiente espacio para el editor, la extensión y los archivos del proyecto.
- Configuraciones de seguridad y privacidad
    - **Autenticación:** Asegúrate de que tu cuenta de GitHub esté correctamente autenticada.
    - **Permisos:** La extensión puede requerir permisos para acceder a los archivos de tu proyecto, así que revisa y ajusta las configuraciones de permisos según sea necesario.
- Actualizaciones
    - **Editor y Extensiones:** Mantén el editor y la extensión de GitHub Copilot actualizados para beneficiarte de las últimas mejoras y correcciones de errores.
- Configuración del proyecto
    - **Lenguaje de Programación:** GitHub Copilot funciona mejor con lenguajes de programación populares y bien soportados como JavaScript, Python, TypeScript, Java, y otros. Asegúrate de que el lenguaje que estás utilizando sea compatible y bien soportado por la extensión.

b. Comparación de rendimiento en diferentes configuraciones de hardware y sistemas operativos.

- CPU:

- **Procesadores Modernos (e.g., Intel Core i5/i7, AMD Ryzen 5/7):**

    - **Rendimiento:** Excelente. Los procesadores modernos proporcionan suficiente potencia para manejar el editor de código, la extensión de Copilot y otras tareas simultáneas sin problemas.
    - **Ventajas:** Mayor velocidad de respuesta, multitarea eficiente.

- **Procesadores Más Antiguos (e.g., Intel Core i3, AMD A-series):**
    - **Rendimiento:** Aceptable, pero puede haber retrasos ocasionales en la respuesta de Copilot, especialmente con proyectos grandes o múltiples extensiones activas.
    - **Ventajas:** Aún funcional, pero puede no ser ideal para cargas de trabajo pesadas.

- RAM:

- **8 GB o Más:**

    - **Rendimiento:** Óptimo. Con suficiente RAM, el sistema puede manejar múltiples extensiones, proyectos grandes y la carga de Copilot sin problemas.

    - **Ventajas:** Reducción de tiempos de carga y mayor capacidad para manejar múltiples tareas.

- **4 GB o Menos:**

    - **Rendimiento:** Adecuado, pero limitado. Puede haber tiempos de respuesta más largos y posibles lentitudes si se ejecutan múltiples aplicaciones o extensiones simultáneamente.

    - **Ventajas:** Aún utilizable, pero puede haber cuellos de botella con tareas exigentes.

c. Consideraciones sobre el consumo de recursos y la eficiencia

- Sistemas Operativos

✓ Windows 10/11:

- **Rendimiento:** Muy bueno. GitHub Copilot y la mayoría de las extensiones están diseñadas para funcionar sin problemas en versiones recientes de Windows.
- **Ventajas:** Amplia compatibilidad con software y buen soporte para actualizaciones.

✓ macOS (Versiones Recientes, e.g., 10.15 Catalina y superior):

- **Rendimiento:** Excelente. GitHub Copilot funciona bien en macOS, con buena integración y soporte en editores como Visual Studio Code.
- **Ventajas:** Integración fluida con el ecosistema Apple y buen rendimiento en hardware moderno.

✓ Linux (Distribuciones Modernas, e.g., Ubuntu 20.04 y superior):

- **Rendimiento:** Bueno. GitHub Copilot es compatible con Linux, pero la experiencia puede variar dependiendo de la distribución y la configuración del sistema.
- **Ventajas:** Flexibilidad y personalización, con buen rendimiento en hardware moderno.

✓ Sistemas operativos más antiguos o menos compatibles:

- **Rendimiento:** Limitado. Los sistemas operativos más antiguos o menos compatibles pueden enfrentar problemas de compatibilidad o rendimiento con GitHub Copilot.
- **Ventajas:** Menor compatibilidad, posibles problemas con actualizaciones y soporte de software.

✓ Resumen de comparación:

- **Hardware moderno (8 GB RAM, SSD, CPU Rápido):**

  - **Mejor Rendimiento:** Respuesta rápida, multitarea eficiente, menor latencia en la sugerencia de código.

- **Hardware moderado (4 GB RAM, HDD, CPU Más Antiguo):**
    - **Rendimiento aceptable:** Puede haber retrasos o lentitud en tareas intensivas, pero sigue siendo funcional.
- **Sistemas operativos recientes y compatibles:**
    - **Mejor rendimiento:** Buena integración y soporte. Funciona bien con GitHub Copilot y otros softwares modernos.
- **Sistemas operativos más antiguos o menos compatibles:**
    - **Rendimiento limitado:** Posibles problemas de compatibilidad y rendimiento reducido.

Bibliografía

[1] E. S. M. Gallart, “Cómo utilizar GitHub Copilot y mejorar el desarrollo de Software - Itequia.” [Online]. Available: https://itequia.com/es/utilizar-github-copilot-mejorar-el-desarrollo/

[2]INGitHub, “Acerca de Copilot en el soporte de GitHub - Documentación de GitHub - GitHub Docs.” [Online]. Available: https://docs.github.com/es/support/learning-about-github-support/about-copilot-in-github-support

[3 Denis Tsyplakov, “GitHub Copilot: lo que se debe y no se debe hacer.” [Online]. Available: https://www.dataart.team/es/articles/github-copilot-dos-and-donts

[4] “Copilot: capacidades, riesgos y desafíos del asistente de IA de Microsoft”. adeprin. Accedido el 20 de julio de 2024. [En línea]. Disponible: https://www.adeprin.org/copilot-capacidades-riesgos-y-desafios-del-asistente-de-ia-de-microsoft/

[5] “Cómo usar Copilot: configuración y aprendizaje de métodos útiles de codificación con IA”. Tutoriales Hostinger. Accedido el 20 de julio de 2024. [En línea]. Disponible: https://www.hostinger.mx/tutoriales/como-usar-copilot#Como_usar_Copilot_para_proporcionar_sugerencias_de_codigo

[6] Ayesa, «Impacto de Copilot en la Productividad y Creatividad en el Trabajo: Un Estudio de Microsoft,» [En línea]. Available: https://tinyurl.com/2dcqv4a4. [Último acceso: s.f].

[7] M. Lopez, «Llevo algunos días usando Copilot de GitHub para programar y esta es mi experiencia,» 2021. [En línea]. Available: https://www.xataka.com/robotica-e-ia/llevo-algunos-dias-usando-copilot-github-para-programar-esta-mi-experiencia.