Universidad Don Bosco

Diseño y programación de software multiplataforma DPS104

Facultad de Ingeniería

Investigación Aplicada 1: Copilot en Visual Studio Code

Grupo teórico: G02T

Docente: Ing. Alexander Alberto Siguenza Campos

Integrantes:

Yensy Alejandra Cruz Barahona	CB121442
Diego Fernando Mancía Hernández	MH212532
Johan Anthony Menjivar Girón	MG182330
Javier Ernesto Pérez Joaquín	PJ211152
Alinson Javier Meléndez Torres	MT191530

Viernes 9 de Agosto de 2024

1. Historia y desarrollo de github copilot

La primera versión de GitHub Copilot fue lanzada inicialmente el 29 de junio de 2021 por su empresa desarrolladora GitHub. Es lanzado como una herramienta integrada en “Visual Studio Code”, que utiliza inteligencia artificial para ayudar a los programadores a la hora de generar fragmentos de código para sus soluciones informáticas.

Esta versión de GitHub Copilot toma como núcleo el modelo de inteligencia artificial “OpenAI Codex” de la empresa OpenAI. Este modelo es un descendiente del modelo GPT-3, el cual fue desarrollado también por OpenAI; sin embargo, a diferencia de GPT-3 que está especializado en comprender y generar lenguaje humano, OpenAI Codex se especializa en comprender y generar código de programación.

Luego del lanzamiento inicial para Visual Studio Code y con OpenIA Codex como núcleo, se amplió la lista de IDEs en los cuales se podría integrar GitHub Copilot; los cuales son:

- En octubre de 2021 es lanzado como complemento en los diferentes IDEs de JetBrains y complemento para Neovim, un editor de texto.
- En marzo de 2022 es lanzado como complemento para Visual Studio 2022.

En último lugar, en marzo 2023 sucedió la última gran actualización de la herramienta GitHub Copilot pasándose a llamar “GitHub Copilot X” el cual como principal cambio pasaba de utilizar OpenIA Codex a utilizar el modelo GPT-4, que es el sucesor directo de GPT-3. Este nuevo modelo de IA tiene una mejor comprensión de instrucciones complejas y una mayor tendencia a errores, por lo que fortalece en gran medida lo ofrecido a los programadores.

Además, GitHub Copilot X agrega las siguientes caracteristicas:

- Un chat integrado que permite interactuar con la herramienta a modo de asistente, recordando a ChatGPT. Cabe mencionar que no se limita solo a texto sino que se puede interactuar por voz, de esta manera GitHub Copilot X

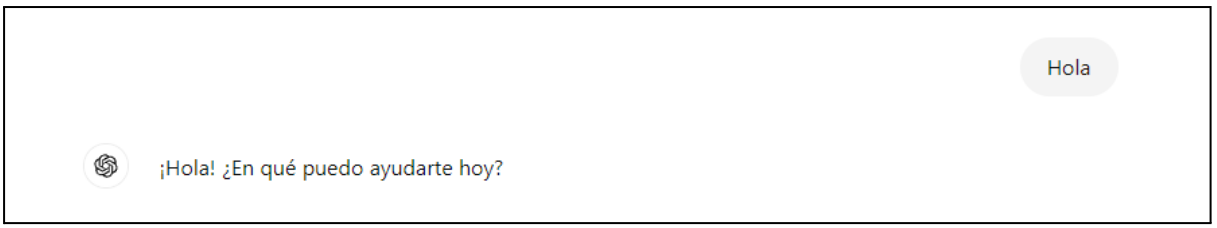
no solo genera fragmentos de código sino que puede hacer sugerencias o identificar errores sobre el código escrito.
- La capacidad de utilizarlo en la CLI (Interfaz de línea de comandos), para poder apoyarnos a la hora de escribir comandos y ejecutarlos.
- Poder generar documentación de un proyecto a partir del código, ya que cuenta con la capacidad de explicar lo que hacen fragmentos de código.

2. Funcionamiento técnico de github copilot

2.1 Conociendo GPT-3

GPT-3 es definido como un modelo de lenguaje autorregresivo o un modelo transformador generativo pre entrenado. Esto quiere decir que es capaz de generar textos que simulan la redacción humana gracias a un periodo de entrenamiento con diferentes fuentes para aprender a comprender el lenguaje humano, se genera texto a partir de instrucciones dadas por el usuario, estas instrucciones tienen que ser escritas y se almacenan en una memoria de contexto para que el modelo “recuerde” sobre que está redactando. Teniendo una versión mejorada denominada como GPT-3.5, que podemos ver su funcionamiento en “ChatGPT” el cuál es una interfaz

que nos permite interactuar con GPT-3.5 y poder escribirle, darle instrucciones, para que nos redacte textos.



Cabe mencionar que este modelo cuenta con la limitante de no poder generar imágenes, vídeos o audios; limitándose únicamente a la redacción de textos.

Para generar sus respuestas el modelo pasó previamente por una etapa de aprendizaje y entrenamiento utilizando diferentes recursos públicos en internet para comprender el lenguaje humano, estos recursos son: Sitios de noticias, Obras literarias, Artículos científicos, etc.

Para procesar la cantidad de recursos usados en su entrenamiento y entender sus interacciones con usuarios, GPT-3 utiliza un modelo “Transformer”, el cual es una arquitectura de red neuronal que consta de dos partes:

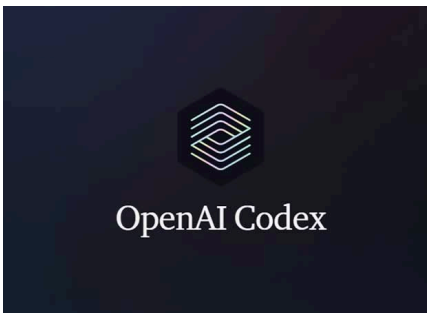
- Encoder: Encargada de recibir una secuencia escrita que funciona como input de entrada y se encarga de procesar cada parte de la secuencia de manera simultánea sin perder la relación entre ellas. Por ejemplo, si la secuencia es una oración analizará cada palabra y la relación entre ellas para comprenderlo.
- Decoder: Se encarga de generar la secuencia de salida a partir de lo “comprendido” por el encoder.

Otro elemento importante de GPT-3 para sus interacciones con los usuarios es su memoria de contexto, que le permite almacenar las partes de las secuencias de entrada y salida que han pasado por el modelo, es decir, almacenar todas las palabras que ha procesado y generado. La memoria de contexto le permite:

- Manejar interacciones prolongadas
- Generar respuestas coherentes en una conversación
- Generar texto basado en secuencias previas

2.2 OpenIA Codex

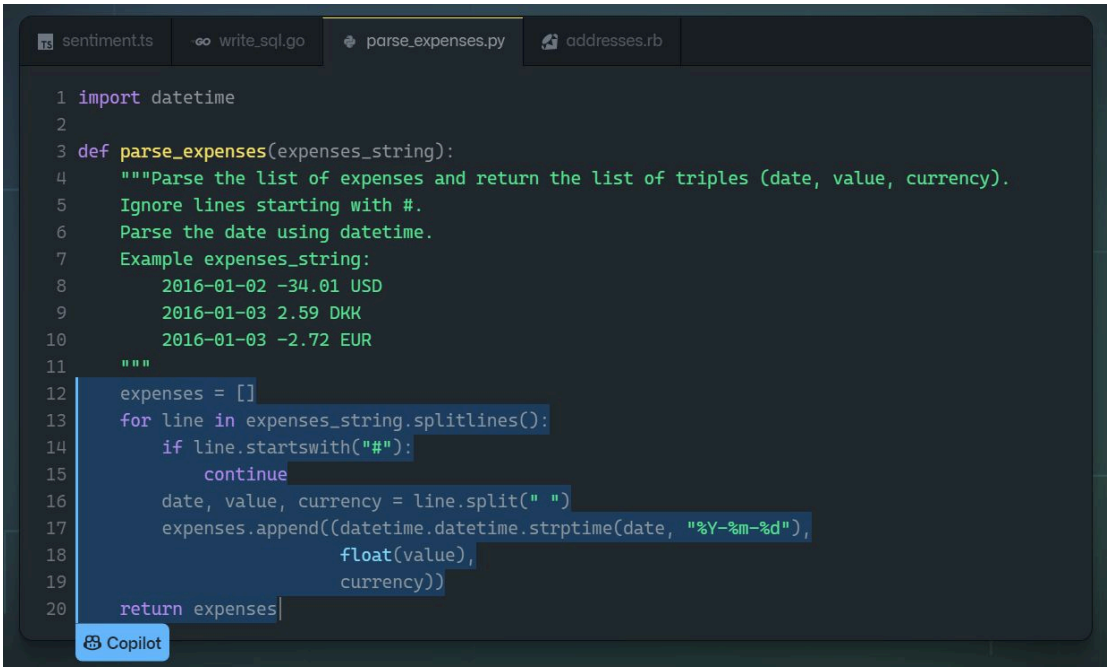
La primera versión de GitHub Copilot utilizaba OpenIA Codex que es definido por la misma OpenIA, empresa encargada de su desarrollo, como un descendiente de GPT-3 enfocado en comprender tareas para generar fragmentos de código que le sean útiles a desarrolladores.



En el apartado técnico conserva el modelo “Transformer” como arquitectura de red neuronal, con la principal diferencia respecto a GPT-3 en el tipo de secuencias de entrada utilizadas en su entrenamiento ya que fue entrenado utilizando el código en repositorios públicos de GitHub para poder generar fragmentos de códigos que resuelvan determinadas tareas. Además, de las secuencias de salida que suele generar ya que GitHub Copilot no genera textos sino fragmentos de código que resuelvan tareas planteadas.

Por otro lado, OpenAI Codex no está pensado para tener una interacción por medio de un Chat sino poder invocarlo en diferentes IDEs con la capacidad de a partir de comentarios con instrucciones de un programador, siendo estas sus secuencias de entrada, generar fragmentos de código, siendo las secuencias de salida, que se pueden implementar de inmediato en proyectos informáticos.

A continuación se presenta un ejemplo, donde se le dio la instrucción a GitHub Copilot de generar una función para formatear una cadena de texto en 3 valores: Fecha, Valor y Moneda; además, de ignorar las cadenas de texto que inicien con “#”.



2.3 GitHub Copilot X y GPT4

GPT-3 pertenece a una familia de modelos de inteligencia artificial, llamada GPT por las siglas de Transformador Generativo Pre Entrenado. Siendo GPT-3 la tercera iteración contando con mayor cantidad de datos utilizados en su entrenamiento y con una mayor memoria de contexto, que sus antecesores.

Sin embargo, GPT-3 fue presentado en febrero de 2020, seguido de una mejora llamada GPT-3.5 la cual es usada en la herramienta de ChatGPT antes de la salida de GPT-4, el modelo sucesor definitivo de GPT-3.

GPT-4 es lanzado en marzo de 2023, utilizando una mayor cantidad de recursos para su período de entrenamiento ya que en esta ocasión el modelo “Transformer” también incluiría recursos con licencias de terceros para su procesamiento y no solo recursos públicos; entre estos recursos se incluye código de programación para tener la capacidad de generar fragmentos de código nativamente y no depender de un descendiente como Codex. Además, cuenta con una ampliación en su memoria de contexto que le permite retener una mayor cantidad de palabras que su antecesor para lograr tener interacciones más largas e incluso más coherencia en las respuestas brindadas.

GitHub Copilot se actualiza con el nombre de GitHub Copilot X, cambiando su modelo de IA de OpenIA Codex a GPT-4 para mejorar los fragmentos de código generados y ampliar las tareas que puede realizar.

GitHub Copilot X es capaz de realizar las siguientes tareas gracias a GPT-4:

- Mantener conversaciones: Al utilizar un modelo pensado para tener conversaciones, Copilot se vuelve un complemento que integra un chat para que el usuario pueda conversar directamente con el asistente y no solo invocarlo en partes del código para que complemento o generes fragmentos de código.
- Interpretar código: Cuenta con la capacidad de procesar fragmentos de código para comprender lo que hace y redactar documentación eficazmente.

- Detectar errores: Al contar con la capacidad de comprender código, también puede apoyar a los programadores detectando errores y proponiendo soluciones.

3. Impacto en la productividad del desarrollador

GitHub Copilot ha demostrado tener un impacto significativo en la productividad de los desarrolladores, según varios estudios y casos prácticos. En investigaciones realizadas por GitHub, se ha encontrado que los desarrolladores que utilizan Copilot pueden escribir código hasta un 40% más rápido en comparación con aquellos que no lo usan. Esto se debe a que Copilot ayuda a reducir el tiempo dedicado a tareas repetitivas y a la generación de código estándar, permitiendo a los desarrolladores centrarse en aspectos más complejos de sus proyectos.

Las opiniones de los desarrolladores que usan Copilot son en su mayoría positivas. Muchos reportan que la herramienta les permite trabajar más rápido y mantener un código más limpio y consistente. Sin embargo, también existen desafíos. Las sugerencias de Copilot no siempre son perfectas y a veces requieren ajustes manuales. Además, hay preocupación sobre la dependencia excesiva en la herramienta, lo que podría afectar el aprendizaje y el desarrollo de habilidades de los nuevos programadores.

4. Ventajas de utilizar github copilot

Ahorro de Tiempo: Genera fragmentos de código rápidamente y reduce tareas repetitivas.

Sugerencias Contextuales: Ofrece sugerencias basadas en el contexto del código.

Soporte para Múltiples Lenguajes: Compatible con diversos lenguajes y frameworks.
Mejora la productividad: Automatiza tareas repetitivas y permite concentrarse en el diseño y la lógica.

Integración con IDEs: Se integra bien con entornos de desarrollo como Visual Studio Code.

Generación de Código de Ejemplo: Ayuda a entender el uso de nuevas bibliotecas o frameworks.

Reducción de Errores: Ofrece sugerencias que siguen buenas prácticas y patrones comunes.

4.1 Casos de uso específicos donde Copilot ha demostrado ser beneficioso

1) **Generación de Código Repetitivo**: Para tareas repetitivas y código repetitivo que no cambia, como la creación de getters y setters en clases, Copilot puede generar estos fragmentos automáticamente, ahorrando tiempo.

2) **Escritura de Funciones y Métodos**: Cuando necesitas implementar funciones o métodos específicos, Copilot puede sugerir implementaciones completas basadas en la firma de la función o el comentario que describes.

3) **Desarrollo de Pruebas Unitarias**: Copilot puede generar automáticamente casos de prueba basados en el código que ya has escrito, facilitando la creación de tests y la cobertura de código.

4) **Integración de APIs**: Facilita el uso de nuevas APIs al proporcionar ejemplos de cómo llamar a funciones o manejar respuestas, lo que es útil al trabajar con bibliotecas o servicios externos.

5. Limitaciones y desafíos de github copilot

GitHub Copilot es una herramienta de autocompletado de código creada por GitHub y OpenAI que ha cambiado la forma del desarrollo de software. Usa inteligencia artificial basada en modelos de lenguaje avanzados, para sugerir fragmentos de código y ayuda en la escritura de programas. Sin embargo, también hay algunas limitaciones que debemos tener en cuenta con su uso.

5.1 Dependencia excesiva y su impacto en el desarrollo de habilidades

Esto ocurre cuando los programadores confían demasiado en la herramienta para escribir su código, en lugar de desarrollar sus propias habilidades de programación, ya que los desarrolladores confían demasiado en las sugerencias automáticas.
Esta dependencia puede afectar de manera negativa el desarrollo de habilidades. Ya que por ejemplo los desarrolladores novatos o principiantes podrían no desarrollar una comprensión profunda del código, porque se acostumbran a aceptar las sugerencias sin cuestionarlas ni comprenderlas completamente.
La falta de práctica en la resolución manual de problemas y en la escritura de código puede limitar el crecimiento profesional y la capacidad de resolver problemas complejos.

5.2 Problemas de Seguridad y Privacidad de Datos

GitHub Copilot genera código a partir de una base de datos masiva de código fuente, que incluye ejemplos de código de repositorios tanto públicos como privados. El código que escribes se envía a los servidores de GitHub y OpenAI para que la herramienta pueda generar sugerencias. Esto podría incluir datos sensibles o propietarios de tus proyectos. Aunque GitHub y OpenAI toman medidas para proteger estos datos, existe el riesgo de que información confidencial pueda ser expuesta o mal gestionada.

El código sugerido por Copilot no siempre está libre de errores o vulnerabilidades de seguridad. Dependiendo de cómo se utilice, puede introducir fallos de seguridad, como vulnerabilidades comunes que los desarrolladores podrían no detectar de inmediato. Esto puede poner en riesgo la seguridad del software final.

5.3 Calidad y Relevancia de las Sugerencias Proporcionadas por Copilot

Aunque Copilot genera sugerencias basadas en el contexto del código, la precisión puede variar y estas sugerencias no siempre pueden ser acertadas. Las sugerencias pueden no ser siempre relevantes o correctas, y pueden necesitar ajustes significativos por parte del desarrollador. Aceptar sugerencias sin revisarlas cuidadosamente puede llevar a que se incluya código que no es eficiente o está mal estructurado, lo que puede afectar la calidad del software en general.

5.4 Reacciones y Críticas de la Comunidad de Desarrolladores

Algunos desarrolladores están preocupados de que el código sugerido pueda ser de baja calidad o no seguir las mejores prácticas. Esto puede llevar a problemas como errores ocultos, vulnerabilidades de seguridad o código que no se ajusta a los estándares del proyecto. Ha habido críticas sobre el uso del código con derechos de autor para entrenar a Copilot.

Algunos desarrolladores están preocupados por cómo Copilot maneja la propiedad intelectual y la autoría del código, temen que el código generado por la herramienta pueda ser demasiado similar al de otros proyectos, lo que podría llevar a problemas de propiedad intelectual y a la posible infracción de derechos de autor.

Algunos argumentan que usar Copilot podría afectar el proceso de aprendizaje de los programadores novatos. Si los principiantes confían en las sugerencias de la herramienta sin entender el código, podrían perder la oportunidad de aprender conceptos fundamentales y desarrollar habilidades de programación de manera más profunda.

Y por último en algunas reacciones positivas los desarrolladores valoran Copilot por su capacidad para aumentar la productividad y reducir el tiempo necesario para escribir código. La herramienta puede ser especialmente útil para tareas repetitivas o para aprender nuevas tecnologías rápidamente.

En conclusión, si bien GitHub Copilot ofrece muchas ventajas, como aumentar la productividad y facilitar el aprendizaje de nuevas tecnologías y beneficios como ahorrar tiempo o productividad para introducir códigos automáticos o autogenerados.

También posee sus limitaciones y es fundamental que los desarrolladores, especialmente los estudiantes o desarrolladores principiantes, usen la herramienta con cuidado. Es necesario evaluar y comprender las sugerencias de la herramienta, para tener la clave de asegurar que el código generado sea de alta calidad y se ajuste a las necesidades del proyecto para el que se está aplicando.

6. Comparación con otras herramientas similares

6.1. Tabnine

- **Descripción:** Tabnine es un asistente de IA que utiliza modelos de aprendizaje automático para predecir y completar código. Ofrece una versión gratuita y opciones premium.
- **Características:**
    - **Integración:** Compatible con múltiples editores, incluyendo Visual Studio Code, IntelliJ IDEA, Sublime Text, y más.
    - **Lenguajes de Programación:** Soporta más de 20 lenguajes de programación.
    - **Uso:** Utilizado para autocompletado de código y predicción de código.
    - **Ventajas:**
        - Soporte para múltiples IDEs y lenguajes.
        - Funciona bien en configuraciones de desarrollo local.
    - **Desventajas:**
        - Las sugerencias pueden ser menos contextuales en comparación con Copilot.
        - La versión gratuita tiene limitaciones en características avanzadas.
    - **Precio:** Ofrece una versión gratuita y una suscripción premium.

6.2. Kite

- **Descripción:** Kite es una herramienta de autocompletado de código que utiliza inteligencia artificial para sugerir fragmentos de código en tiempo real.

- **Características:**
    - **Integración:** Compatible con editores como VS Code, PyCharm, Atom, y más.
    - **Lenguajes de Programación:** Principalmente enfocado en Python, pero también soporta otros lenguajes como JavaScript, Java, C++, entre otros.
    - **Uso:** Ofrece completado de líneas de código y documentación dentro del editor.
    - **Ventajas:**
        - Funciona sin conexión después de la instalación.
        - Su enfoque en Python lo hace especialmente poderoso para desarrolladores en este lenguaje.
    - **Desventajas:**
        - Menos compatible con lenguajes fuera de su enfoque principal.
        - Requiere instalación local que puede consumir recursos.
    - **Precio:** Tiene una versión gratuita y otra de pago con características avanzadas.

6.3. Codeium

- **Descripción:** Codeium es una herramienta gratuita que proporciona autocompletado y sugerencias de código impulsadas por IA.
- **Características:**
    - **Integración:** Disponible en varios IDEs y editores.
    - **Lenguajes de Programación:** Soporta una amplia gama de lenguajes, desde los más populares hasta algunos más especializados.
    - **Uso:** Diseñado para sugerir líneas de código y funciones, y ayudar a los desarrolladores a escribir código más rápido.
    - **Ventajas:**
        - Totalmente gratuita.
        - Ofrece una experiencia de autocompletado fluida y rápida.
    - **Desventajas:**
        - Todavía en desarrollo activo, puede carecer de algunas características de herramientas más maduras.
    - **Precio:** Completamente gratuita.

6.4. Replit Ghostwriter

- **Descripción:** Ghostwriter es una herramienta de IA integrada en la plataforma de desarrollo Replit, diseñada para ayudar a escribir código con mayor eficiencia.
- **Características:**
    - **Integración:** Exclusivo para la plataforma Replit.

- **Lenguajes de Programación:** Compatible con varios lenguajes de programación disponibles en Replit.
- **Uso:** Ofrece autocompletado y sugerencias contextuales de código dentro del entorno Replit.
- **Ventajas:**
  - Integración profunda con Replit.
  - Buen soporte para proyectos colaborativos.
- **Desventajas:**
  - Limitado a los usuarios de Replit.
  - Menos flexible fuera del entorno Replit.
- **Precio:** Requiere suscripción a Replit Pro para el acceso completo.

7. Requerimientos de hardware y software

Visual studio code es amigable en computadoras de hardware limitado, ahora bien, la extensión de github copilot puede agregarle cierto tipo de carga a los dispositivos por lo que es recomendable tomar en cuenta este aumento de requerimientos.

Los requisitos mínimos y recomendados para poder correr sin problemas visual studio con github copilot son los siguientes:

Requisitos mínimos:

Procesador: Procesador de 1.6 GHz o superior.

Espacio de almacenamiento: 2Gb libre.

RAM: 1 Gb.

Sistema operativo: Windows 10, 11, macOS, Linux(Debian) o Linux(Red Hat).

GPU dedicada: No necesaria.

Requisitos recomendados:

Procesador: Procesador de 2 GHz o superior.

Espacio de almacenamiento: 2Gb libre.

RAM: 4 Gb.

Sistema operativo: Windows 10, 11, macOS, Linux (Debian) o Linux (Red Hat).

GPU dedicada: No necesaria.

En cuanto al sistema operativo, no representa mayor diferencia en cuanto a rendimiento se refiere debido a la naturaleza de visual studio code las comparativas nos darían valores casi exactos de rendimiento entre los diferentes sistemas operativos mencionados en los requerimientos asumiendo que se usa computadoras con hardware similar.

En cuanto a la eficiencia de la aplicación visual studio junto con copilot es de tomar en cuenta las siguientes consideraciones:

- Las demás extensiones que se tengan instaladas pueden generar un impacto en el rendimiento de la aplicación o puede que incluso interfieran con algunas funciones de copilot
- Los framework a utilizar o la carga de trabajo a la que se somete el pc al momento de ejecutar pruebas puede reducir el rendimiento temporalmente, para ello es importante contar con un equipo lo suficientemente potente para poder gestionar varias cargas a la vez.