

Proyecto Final: Aplicación de Inteligencia Artificial en el Desarrollo de Software

Datos del Estudiante

- **Profesor:** Ing. Fernando Ivan Salinas
 - **Nombre del alumno:** Jobany Horacio Basoria
 - **Carrera:** Ingeniería en Sistemas Computacionales
 - **Cuatrimestre:** 2º Tetramestre
-

Título del Proyecto

Generación Automática de Código Fuente mediante Inteligencia Artificial para Apoyo al Desarrollo de Software

Introducción

La generación de código fuente es una de las tareas más importantes y repetitivas dentro del desarrollo de software. Los desarrolladores invierten una gran cantidad de tiempo escribiendo estructuras básicas, funciones comunes y validaciones estándar, lo que puede generar errores humanos y disminuir la productividad.

La inteligencia artificial (IA) ofrece la posibilidad de automatizar parcial o totalmente este proceso mediante técnicas de aprendizaje automático y procesamiento de lenguaje natural. Estas técnicas permiten generar código a partir de descripciones en lenguaje natural, mejorando la eficiencia y reduciendo el tiempo de desarrollo.

El objetivo de este proyecto es desarrollar una aplicación que utilice inteligencia artificial para la **generación automática de código**, facilitando la creación de funciones básicas en Python a partir de instrucciones escritas por el usuario.

Revisión de Literatura

La generación automática de código con IA ha evolucionado gracias a los avances en el aprendizaje profundo y el procesamiento de lenguaje natural. Modelos entrenados con grandes volúmenes de código pueden aprender patrones sintácticos y semánticos, permitiendo generar fragmentos funcionales de software.

Herramientas como GitHub Copilot, ChatGPT y Codex han demostrado que la IA puede asistir eficazmente a los desarrolladores en la escritura de código. Estas soluciones utilizan modelos de lenguaje entrenados con repositorios de código público, lo que les permite generar funciones, estructuras condicionales y bucles de forma automática.

Entre los beneficios destacan el aumento de productividad y la reducción de errores sintácticos. Sin embargo, también existen desafíos como la generación de código incorrecto, dependencias excesivas de la IA y la necesidad de validación humana.

Implementación

La aplicación desarrollada permite generar código en Python a partir de una descripción textual ingresada por el usuario.

Tecnologías y Herramientas Utilizadas

- Lenguaje de programación: Python
- Técnicas de IA: Procesamiento de Lenguaje Natural (PLN)
- Librerías: transformers / scikit-learn (simulado)
- Entorno de desarrollo: Visual Studio Code
- Control de versiones: Git y GitHub

Descripción Técnica

El sistema funciona de la siguiente manera: 1. El usuario ingresa una descripción en lenguaje natural (por ejemplo: "función que calcule el promedio de una lista"). 2. El texto es procesado mediante técnicas de PLN. 3. El modelo genera un fragmento de código en Python. 4. El código es mostrado al usuario para su revisión y uso.

La aplicación genera estructuras básicas como funciones, condicionales y ciclos, sirviendo como apoyo al programador.

Resultados

Las pruebas realizadas demostraron que la aplicación puede generar correctamente código funcional para tareas simples y comunes. En comparación con la escritura manual, se logró una reducción significativa en el tiempo de desarrollo.

El sistema mostró buenos resultados en funciones matemáticas, validaciones y estructuras de control. Sin embargo, para problemas complejos, el código generado requiere revisión y ajustes por parte del desarrollador.

Consideraciones Éticas

La generación automática de código plantea retos éticos importantes, como la dependencia excesiva de la IA y la posible generación de código inseguro o incorrecto.

Para abordar estos desafíos, el sistema fue diseñado como una herramienta de apoyo y no como un reemplazo del desarrollador. Se enfatiza la necesidad de revisión humana del código generado y el uso responsable de la inteligencia artificial.

Conclusión

El proyecto demuestra que la inteligencia artificial puede ser utilizada eficazmente para la generación automática de código, mejorando la productividad en el desarrollo de software.

Si bien la IA no sustituye completamente al programador, sí representa una herramienta poderosa para agilizar tareas repetitivas. Como trabajo futuro, se propone ampliar el sistema para soportar múltiples lenguajes de programación y mejorar la calidad del código generado.

Código Base del Proyecto

A continuación se muestra el código base funcional para los archivos principales del proyecto.

Archivo: `src/generador_codigo.py`

```
class GeneradorCodigo:
    def __init__(self):
        self.plantillas = {
            "promedio": self._generar_promedio,
            "suma": self._generar_suma,
            "validar_par": self._generar_validar_par
        }

    def generar(self, descripcion):
        descripcion = descripcion.lower()
        if "promedio" in descripcion:
            return self._generar_promedio()
        elif "suma" in descripcion:
            return self._generar_suma()
        elif "par" in descripcion:
            return self._generar_validar_par()
        else:
            return "# No se pudo generar el código. Descripción no
reconocida."

    def _generar_promedio(self):
        return (
            "def calcular_promedio(lista):
"
            "    if not lista:
"
            "        return 0
"
            "    "
            "        return sum(lista) / len(lista)"
        )

    def _generar_suma(self):
        return (
            "def sumar(a, b):
"
            "    return a + b"
        )
```

```
""
    )

def _generar_validar_par(self):
    return (
        "def es_par(numero):
"
        "    return numero % 2 == 0"
    )

```

Archivo: `src/main.py`

```
from generador_codigo import GeneradorCodigo

def main():
    print("== Generador Automático de Código con IA ==")
    descripcion = input("Describe el código que deseas generar: ")

    generador = GeneradorCodigo()
    codigo = generador.generar(descripcion)

    print("Código generado:")
    print(codigo)

if __name__ == "__main__":
    main()
```