**UNIVERSIDAD MARIANO GÁLVEZ DE GUATEMALA**

**SEDE ANTIGUA GUATEMALA**

**FACULTAD DE INGENIERÍA EN SISTEMAS**

**CURSO: INTELIGENCIA ARTIFICIAL**

**Segundo Parcial - Laboratorio**

**Calendario

Descripción generada automáticamente**

**Horacio Lopez: 1290-21-3372**

**CATEDRÁTICO**

**ING. JOSUE MAGDALENO FLORIAN ARRIAZA**

**Fecha: 07/05/2025**

**entrenar\_modelo.py**

Texto

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.Este archivo se encarga de entrenar al modelo de inteligencia artificial que clasifica lo que el usuario escribe. Básicamente, aquí definimos varias frases relacionadas con temas de ciberseguridad (como phishing, contraseñas inseguras, malware, entre otros.) y las etiquetamos según su categoría. Luego, usamos una técnica de PLN llamada TF-IDF para convertir ese texto en números que una computadora puede entender, y con eso entrenamos un modelo de aprendizaje automático (Naive Bayes). Al final, el modelo se guarda en un archivo llamado modelo.pkl, que después usaremos para que el agente pueda “entender” nuevas frases que no ha visto antes.

**main.py**

Texto

El contenido generado por IA puede ser incorrecto.Texto

El contenido generado por IA puede ser incorrecto.Este es el archivo principal que ejecuta al agente inteligente. Aquí es donde el usuario puede escribir libremente su problema o duda, y el agente intenta detectar de qué tema se trata usando el modelo entrenado previamente. Si el modelo lo reconoce como un tema de ciberseguridad, el agente busca una causa y una recomendación en una base de datos PostgreSQL, y responde de forma amigable. Además, si el agente no está seguro de que el texto tenga que ver con ciberseguridad, lo dice claramente. También incluye preguntas de seguimiento y frases variadas para que la conversación se sienta más natural.

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.

**Librerías utilizadas:**

* **pickle:** Se usa para guardar y cargar el modelo de machine learning que entrenamos. Gracias a esto, no hay que volver a entrenarlo cada vez que usamos el agente.
* **scikit-learn (sklearn):** Es la librería principal para todo el proceso de Machine Learning. Usamos TfidfVectorizer para convertir texto en números (PLN) y MultinomialNB para entrenar un modelo que clasifica frases según su tema.
* **psycopg2:** Es la librería que conecta Python con la base de datos PostgreSQL. Nos permite hacer consultas a la base para buscar problemas, causas y recomendaciones.
* **random:** Se usa para que el agente elija frases y respuestas de forma más natural y no siempre diga lo mismo, haciendo la conversación más realista.

**Conceptos utilizados**

**Procesamiento de Lenguaje Natural (PLN):**

El usuario puede escribir frases libremente, y el agente usa TfidfVectorizer para transformar esas frases en vectores numéricos que el modelo puede entender. Este paso permite que el agente “interprete” lo que el usuario quiso decir, incluso si no usa las mismas palabras exactas que vimos en el entrenamiento.

**Machine Learning (Aprendizaje Automático):**

Entrenamos un modelo con ejemplos reales de frases clasificadas en categorías como phishing, contraseñas, etc. Luego, el modelo aprende patrones y puede clasificar nuevas frases automáticamente. Esto es lo que permite que el agente se adapte a distintas entradas del usuario sin usar if fijos.

**Base de conocimiento (Base de datos PostgreSQL):**

Una vez el agente detecta el tema de la frase, consulta una tabla de hechos en PostgreSQL donde están guardados problemas comunes, causas y recomendaciones por categoría. Así, el agente responde con información precisa y útil.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

**Inferencia lógica:**

Si bien no usamos lógica simbólica como “P → Q”, el agente sí aplica una forma simple de inferencia: si detecta que el problema pertenece a una categoría (por ejemplo, phishing), entonces busca una causa y una recomendación lógica para ese tipo de problema. Se basa en la regla de Modus Ponens: “Si el usuario dijo X, entonces probablemente se trata de Y, por lo tanto, la solución es Z.” o bien “Si P entonces Q, por lo tanto, P”