**Código para deepseek**

import json

import random

import requests

import re

import tkinter as tk

from tkinter import scrolledtext

# Configuración del endpoint de LM Studio

LM\_STUDIO\_API\_URL = "http://localhost:1234/v1/chat/completions"

# Cargar las preguntas desde el archivo JSON

def cargar\_preguntas(archivo="Factorizacion\_dif\_cuadrados.json"):

with open(archivo, 'r', encoding='utf-8') as f:

return json.load(f)["preguntas"]

# Obtener preguntas aleatorias

def obtener\_preguntas\_aleatorias(preguntas, cantidad=2):

return random.sample(preguntas, min(cantidad, len(preguntas)))

# Limpieza robusta de la respuesta del modelo

def limpiar\_respuesta\_modelo(texto):

texto = re.sub(r"<think>.\*?</think>", "", texto, flags=re.DOTALL)

texto = re.sub(r"\\[a-zA-Z]+\b", "", texto)

texto = re.sub(r"\$+", "", texto)

texto = re.sub(r"\\\(|\\\)", "", texto)

texto = re.sub(r"\\", "", texto)

texto = re.sub(r"[\[\]]", "", texto)

texto = texto.replace("\*\*", "").replace("\*", "")

texto = re.sub(r"#+", "", texto)

return texto.strip()

# Generar explicaciones desde LM Studio

def generar\_explicacion\_local(errores):

if not errores:

return "¡Bien hecho! Parece que dominas la factorización por diferencia de cuadrados. Sigue practicando con ejercicios más avanzados."

errores\_formateados = "\n".join(

[f"Se cometió un error en la pregunta {num + 1}: {preguntas\_seleccionadas[num]['pregunta']}. "

f"Respuesta dada: {resp}" for num, resp in errores]

)

prompt = f"""

Un estudiante cometió errores en las siguientes preguntas de factorización por diferencia de cuadrados:

{errores\_formateados}

Por favor:

- No uses comandos de LaTeX, ni ecuaciones entre \\( \\), ni símbolos como \\.

- No uses Markdown.

- Escribe todo en texto plano.

- Comienza con "Se cometió un error en la pregunta X: [Texto de la pregunta]".

- Luego describe el error y cómo debió resolverse.

- Da la respuesta correcta.

- Añade consejos prácticos para evitar errores similares.

Responde completamente en español.

"""

data = {

"model": "deepseek-r1-distill-llama-8b",

"messages": [

{"role": "system", "content": "Eres un tutor de matemáticas experto en factorización."},

{"role": "user", "content": prompt}

],

"temperature": 0.7

}

response = requests.post(LM\_STUDIO\_API\_URL, json=data)

if response.status\_code == 200:

respuesta\_modelo = response.json()["choices"][0]["message"]["content"]

return limpiar\_respuesta\_modelo(respuesta\_modelo)

else:

return "⚠️ No se pudo conectar con el modelo local. Verifica que LM Studio está ejecutándose."

# Normalizar respuesta para comparación

def normalizar\_respuesta(respuesta):

return sorted(respuesta.replace(" ", "").lower().strip("()").split(")("))

# Mostrar recomendaciones en el área de texto

def mostrar\_recomendaciones():

recomendaciones = generar\_explicacion\_local(errores)

recomendacion\_text.config(state=tk.NORMAL)

recomendacion\_text.delete(1.0, tk.END)

recomendacion\_text.insert(tk.END, recomendaciones)

recomendacion\_text.config(state=tk.DISABLED)

recomendacion\_text.pack(padx=15, pady=10, fill="both", expand=True)

# Evaluar respuesta individual

def evaluar\_respuesta(index):

global puntaje\_total

respuesta\_usuario = entradas\_respuestas[index].get().strip()

respuesta\_correcta = preguntas\_seleccionadas[index]["respuesta"]

if normalizar\_respuesta(respuesta\_usuario) == normalizar\_respuesta(respuesta\_correcta):

etiquetas\_resultados[index].config(text="✅ Correcto", fg="green")

puntaje\_total += 1

else:

etiquetas\_resultados[index].config(text=f"❌ Incorrecto. Respuesta: {respuesta\_correcta}", fg="red")

errores.append((index, respuesta\_usuario))

entradas\_respuestas[index].config(state=tk.DISABLED)

botones\_evaluar[index].config(state=tk.DISABLED)

if all(boton["state"] == tk.DISABLED for boton in botones\_evaluar):

mostrar\_puntaje()

# Mostrar puntaje final

def mostrar\_puntaje():

total\_preguntas = len(preguntas\_seleccionadas)

porcentaje = (puntaje\_total / total\_preguntas) \* 100

etiqueta\_puntaje.config(text="Puntaje final:")

resultado\_final.config(text=f" {porcentaje:.2f}%", fg="blue")

frame\_puntaje.pack(pady=5)

boton\_recomendaciones.pack(side="left", padx=10)

# Interfaz gráfica

ventana = tk.Tk()

ventana.title("Evaluación de Álgebra")

ventana.geometry("850x550")

# Cargar preguntas

preguntas = cargar\_preguntas()

preguntas\_seleccionadas = obtener\_preguntas\_aleatorias(preguntas, 2)

puntaje\_total = 0

errores = []

# Widgets

titulo = tk.Label(ventana, text="Evaluación Factorización por Diferencia de Cuadrados", font=("Arial", 13, "bold"))

titulo.pack(pady=10)

frame\_preguntas = tk.Frame(ventana)

frame\_preguntas.pack()

pregunta\_labels = []

entradas\_respuestas = []

botones\_evaluar = []

etiquetas\_resultados = []

for i, pregunta in enumerate(preguntas\_seleccionadas):

frame = tk.Frame(frame\_preguntas)

frame.pack(pady=5)

pregunta\_label = tk.Label(frame, text=f"{i + 1}. {pregunta['pregunta']}", font=("Arial", 12, "bold"))

pregunta\_label.pack(side="left", padx=5)

entrada\_respuesta = tk.Entry(frame, width=20, font=("Arial", 12))

entrada\_respuesta.pack(side="left", padx=5)

boton\_evaluar = tk.Button(frame, text="Evaluar", command=lambda i=i: evaluar\_respuesta(i), font=("Arial", 12, "bold"), bg="blue", fg="white")

boton\_evaluar.pack(side="left", padx=5)

etiqueta\_resultado = tk.Label(frame\_preguntas, text="", font=("Arial", 12, "bold"))

etiqueta\_resultado.pack()

pregunta\_labels.append(pregunta\_label)

entradas\_respuestas.append(entrada\_respuesta)

botones\_evaluar.append(boton\_evaluar)

etiquetas\_resultados.append(etiqueta\_resultado)

# Puntaje y recomendaciones

frame\_puntaje = tk.Frame(ventana)

frame\_puntaje.pack(pady=5)

etiqueta\_puntaje = tk.Label(frame\_puntaje, text="", font=("Arial", 12, "bold"))

etiqueta\_puntaje.pack(side="left")

resultado\_final = tk.Label(frame\_puntaje, text="", font=("Arial", 13, "bold"))

resultado\_final.pack(side="left", padx=10)

boton\_recomendaciones = tk.Button(frame\_puntaje, text="Mostrar Recomendaciones", command=mostrar\_recomendaciones, font=("Arial", 12, "bold"), bg="green", fg="white")

boton\_recomendaciones.pack\_forget()

recomendacion\_text = scrolledtext.ScrolledText(ventana, wrap=tk.WORD, width=90, height=10, font=("Arial", 12))

recomendacion\_text.pack\_forget()

ventana.mainloop()