Documentation for edu_tutor_ai.py

This document provides documentation for the `edu_tutor_ai.py` script. The script implements an AI-powered educational assistant using Hugging Face Transformers, PyTorch, and Gradio. It provides two main features: 1. Concept Explanation – generates detailed explanations for given concepts. 2. Quiz Generator – creates quizzes with multiple types of questions and answers.

Dependencies

The script requires the following libraries: - transformers - torch - gradio

Model Initialization

The script loads the IBM Granite model (`ibm-granite/granite-3.2-2b-instruct`) and sets up a tokenizer and model. It automatically detects CUDA availability and adjusts settings accordingly.

Functions

1. generate_response(prompt, max_length=512): - Encodes the input prompt and generates a text response using the model. - Supports GPU acceleration if available. 2. concept_explanation(concept): - Generates a detailed explanation of the given concept with examples. 3. quiz_generator(concept): - Generates 5 quiz questions of various types based on the given concept. - Provides answers in a separate ANSWERS section.

User Interface

The script uses Gradio's `Blocks` interface with two tabs: - Concept Explanation: User enters a concept, and the assistant generates an explanation. - Quiz Generator: User enters a topic, and the assistant generates a quiz with answers.

Execution

The app is launched with Gradio's `.launch(share=True)`, making it accessible via a shareable link.

Source Code

The full source code of the script is included below for reference:

```
# -*- coding: utf-8 -*-
"""EDU TUTOR AI.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/181H9twa1vTT3mUOp6Ffn4JXKgL5zyl23
"""

!pip install transformers torch gradio -q
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

# Load model and tokenizer
model_name = "ibm-granite/granite-3.2-2b-instruct"
```

```
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
    device_map="auto" if torch.cuda.is_available() else None
)

if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token

def generate_response(prompt, max_length=512):
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)

    if torch.cuda.is_available():
        inputs = {k: v.to(model.device) for k, v in inputs.items()}

    with torch.no_grad():
        outputs = model.generate(
            **inputs,
            max_length=max_length,
            temperature=0.7,
            do_sample=True,
            pad_token_id=tokenizer.eos_token_id
        )

    response = tokenizer.decode(outputs[0], skip_special_tokens=True)
    response = response.replace(prompt, "").strip()
    return response

def concept_explanation(concept):
    prompt = f"Explain the concept of {concept} in detail with examples:"
    return generate_response(prompt, max_length=800)

def quiz_generator(concept):
    prompt = f"Generate 5 quiz questions about {concept} with different question types (multiple choice, true/fa
    return generate_response(prompt, max_length=1000)

# Create Gradio interface
with gr.Blocks() as app:
    gr.Markdown("# Educational AI Assistant")

    with gr.Tabs():
        with gr.TabItem("Concept Explanation"):
            concept_input = gr.Textbox(label="Enter a concept", placeholder="e.g., machine learning")
            explain_btn = gr.Button("Explain")
            explanation_output = gr.Textbox(label="Explanation", lines=10)

            explain_btn.click(concept_explanation, inputs=concept_input, outputs=explanation_output)

        with gr.TabItem("Quiz Generator"):
            quiz_input = gr.Textbox(label="Enter a topic", placeholder="e.g., physics")
            quiz_btn = gr.Button("Generate Quiz")
            quiz_output = gr.Textbox(label="Quiz Questions", lines=15)

            quiz_btn.click(quiz_generator, inputs=quiz_input, outputs=quiz_output)

app.launch(share=True)
```