Code Documentation

This document provides a detailed overview of the uploaded Python script. The script is designed to generate eco-friendly tips and summarize policy documents using a language model. It leverages Hugging Face Transformers, Gradio for the UI, and PyPDF2 for PDF processing.

1. Libraries Used

- **gradio**: For building the web interface. - **torch**: For handling deep learning computations. - **transformers**: To load and use pre-trained language models. - **PyPDF2**: For reading and extracting text from PDF documents. - **io**: For handling file streams.

2. Model Loading

The script loads the 'ibm-granite/granite-3.2-2b-instruct' model and tokenizer from Hugging Face. It automatically uses GPU if available, otherwise defaults to CPU.

3. Core Functions

- **generate_response(prompt, max_length)**: Generates a text response from the model. - **extract_text_from_pdf(pdf_file)**: Extracts text from uploaded PDFs. - **eco_tips_generator(problem_keywords)**: Produces actionable eco-friendly tips. - **policy_summarization(pdf_file, policy_text)**: Summarizes policy documents.

4. Gradio Interface

The UI is built using Gradio and consists of two main tabs: - **Eco Tips Generator Tab**: Users enter environmental keywords to get sustainable tips. - **Policy Summarization Tab**: Users upload a PDF or paste text to get a summary and key points.

5. Full Source Code

```
# -*- coding: utf-8 -*- """Untitled0.ipynb Automatically generated by Colab.
Original file is located at
https://colab.research.google.com/drive/1HyJcg252Rz2OnpvCyO8da0Tky7nR4K71 """ !pip
install transformers torch gradio PyPDF2 -q !pip install transformers torch gradio
PyPDF2 -q import gradio as gr import torch from transformers import AutoTokenizer,
AutoModelForCausalLM import PyPDF2 import io # 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=1024): 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 extract_text_from_pdf(pdf_file): if pdf_file is None: return "" try:
pdf_reader = PyPDF2.PdfReader(pdf_file) text = "" for page in pdf_reader.pages: text
+= page.extract_text() + "\n" return text except Exception as e: return f"Error
reading PDF: {str(e)}" def eco_tips_generator(problem_keywords): prompt = f"Generate
practical and actionable eco-friendly tips for sustainable living related to:
{problem_keywords}. Provide specific solutions and suggestions:" return
generate_response(prompt, max_length=1000) def policy_summarization(pdf_file,
```

```
policy_text): # Get text from PDF or direct input if pdf_file is not None: content =
extract_text_from_pdf(pdf_file) summary_prompt = f"Summarize the following policy
document and extract the most important points, key provisions, and
implications:\n\n{content}" else: summary_prompt = f"Summarize the following policy
document and extract the most important points, key provisions, and
implications:\n\n{policy_text}" return generate_response(summary_prompt,
max_length=1200) # Create Gradio interface with gr.Blocks() as app: gr.Markdown("#
Eco Assistant & Policy Analyzer") with gr.Tabs(): with gr.TabItem("Eco Tips
Generator"): with gr.Row(): with gr.Column(): keywords_input = gr.Textbox(
label="Environmental Problem/Keywords", placeholder="e.g., plastic, solar, water
waste, energy saving...", lines=3 ) generate_tips_btn = gr.Button("Generate Eco
Tips") with gr.Column(): tips_output = gr.Textbox(label="Sustainable Living Tips",
lines=15) generate_tips_btn.click(eco_tips_generator, inputs=keywords_input,
outputs=tips_output) with gr.TabItem("Policy Summarization"): with gr.Row(): with
gr.Column(): pdf_upload = gr.File(label="Upload Policy PDF", file_types=[".pdf"])
policy_text_input = gr.Textbox( label="Or paste policy text here",
placeholder="Paste policy document text...", lines=5 ) summarize_btn =
gr.Button("Summarize Policy") with gr.Column(): summary_output =
gr.Textbox(label="Policy Summary & Key Points", lines=20)
summarize_btn.click(policy_summarization, inputs=[pdf_upload, policy_text_input],
outputs=summary_output) app.launch(share=True)
```