```
!pip install transformers torch gradio PyPDF2 -q
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 232.6/232.6 kB 6.0 MB/s eta 0:00:00
```

```
mport gradio as gr
mport torch
om transformers import AutoTokenizer, AutoModelForCausalLM
mport PyPDF2
mport io

 Load model and tokenizer
odel_name = "ibm-granite/granite-3.2-2b-instruct"
okenizer = AutoTokenizer.from_pretrained(model_name)
odel = 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


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

ef 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

ef 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)}"

ef eco_tips_generator(problem_keywords):
    prompt = f"Generate practical and actionable eco-friendly tips for sustainable living related to:
    return generate_response(prompt, max_length=1000)
```

```
ef 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 poi
   else:
       summary_prompt = f"Summarize the following policy document and extract the most important poi

   return generate_response(summary_prompt, max_length=1200)

 Create Gradio interface
ith 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

op.launch(share=True)
```

```
usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
he secret `HF_TOKEN` does not exist in your Colab secrets.
o authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggin
ou will be able to reuse this secret in all of your notebooks.
lease note that authentication is recommended but still optional to access public models or da
 warnings.warn(
okenizer_config.json:        8.88k/? [00:00<00:00, 871kB/s]
ocab.json:        777k/? [00:00<00:00, 17.2MB/s]
erges.txt:        442k/? [00:00<00:00, 24.3MB/s]
okenizer.json:        3.48M/? [00:00<00:00, 90.2MB/s]
dded_tokens.json: 100%                                                  87.0/87.0 [00:00<00:00, 6.93kB/s]
pecial_tokens_map.json: 100%                                            701/701 [00:00<00:00, 57.5kB/s]
onfig.json: 100%                                                        786/786 [00:00<00:00, 94.1kB/s]
torch_dtype` is deprecated! Use `dtype` instead!
odel.safetensors.index.json:        29.8k/? [00:00<00:00, 1.99MB/s]
etching 2 files: 100%                                                   2/2 [01:18<00:00, 78.35s/it]
odel-00002-of-00002.safetensors: 100%                                   67.1M/67.1M [00:01<00:00, 66.7MB/s]
odel-00001-of-00002.safetensors: 100%                                   5.00G/5.00G [01:18<00:00, 204MB/s]
oading checkpoint shards: 100%                                          2/2 [00:17<00:00,  7.37s/it]
eneration_config.json: 100%                                             137/137 [00:00<00:00, 7.81kB/s]
olab notebook detected. To show errors in colab notebook, set debug=True in launch()
 Running on public URL: https://624244601e3c6c35d1.gradio.live

his share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio dep
```

Eco Assistant & Policy Analyzer

Eco Tips Generator | Policy Summarization

Environmental Problem/Keywords

e.g., plastic, solar, water waste, energy saving...

Generate Eco Tips

Sustainable Living Tips