

```

!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
    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, )

    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 {problem_keywords}"
    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 provide actionable steps: {content}"
    else:
        summary_prompt = f"Summarize the following policy document and provide actionable steps: {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",
                        lines=3
                    )
                    generate_tips_btn = gr.Button("Generate Eco Tips")

                with gr.Column():
                    tips_output = gr.Textbox(label="Sustainable Living Tips")

            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")

            summarize_btn.click(policy_summarization, inputs=[pdf_upload, policy_text_input], outputs=summary_output)

```

```
app.launch(share=True)
```

```
232.6/232.6 kB 3.3 MB/s
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: L
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings t
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access
warnings.warn(

tokenizer_config.json:      8.88k/? [00:00<00:00, 324kB/s]

vocab.json:      777k/? [00:00<00:00, 8.45MB/s]

merges.txt:      442k/? [00:00<00:00, 13.3MB/s]

tokenizer.json:      3.48M/? [00:00<00:00, 45.7MB/s]

added_tokens.json: 100%                                87.0/87.0 [00:00<00:00, 6.68kB/s]

special_tokens_map.json: 100%                          701/701 [00:00<00:00, 41.0kB/s]

config.json: 100%                                       786/786 [00:00<00:00, 52.1kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json:      29.8k/? [00:00<00:00, 2.34MB/s]
```

Start coding or [generate](#) with AI.

```
00002.safetensors: 100%

model-00002-of-                                67.1M/67.1M [00:06<00:00, 9.49MB/s]

00002.safetensors: 100%

Loading checkpoint shards: 100%                    2/2 [00:27<00:00, 11.47s/it]

generation_config.json: 100%                      137/137 [00:00<00:00, 12.0kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in
* Running on public URL: https://6faa68c0a503dccee0.gradio.live
```

This share link expires in 1 week. For free permanent hosting and GPU upgrad

## Eco Assistant & Policy Analyzer

Eco Tips Generator

Policy Summarization

Environmental Problem/Keywords

e.g. plastic solar water waste energy saving