Sustainable Smart City Assistance - Detailed Documentation

# Introduction

The Sustainable Smart City Assistance application is an AI-powered solution designed to promote eco-friendly practices and assist in understanding government policies related to sustainability. The system utilizes the IBM Granite language model to generate useful tips for sustainable living and to analyze policy documents by providing clear and concise summaries. By combining Natural Language Processing (NLP) with a user-friendly interface built in Gradio, the application makes advanced AI capabilities accessible to city planners, policymakers, and citizens.

This document provides an in-depth explanation of the codebase, including installation, model loading, functionality of each method, and details of the Gradio interface. Examples and usage scenarios are included to illustrate how the system can be applied in real-world contexts.

# Installation and Setup

Before running the application, ensure that the required dependencies are installed. The script begins by installing the following libraries:  
  
- transformers: Provides access to pretrained language models and tokenizers.  
- torch: The PyTorch deep learning framework used to run the model efficiently.  
- gradio: A Python library for building web-based user interfaces for machine learning applications.  
- PyPDF2: A library used for reading and extracting text from PDF documents.  
  
The installation command used in the script:  
!pip install transformers torch gradio PyPDF2 -q  
  
Ensure that Python 3.8 or higher is installed along with pip to manage dependencies.

# Model Loading

The application loads the IBM Granite model `ibm-granite/granite-3.2-2b-instruct` from Hugging Face. This model is designed for instruction-following tasks, making it suitable for generating eco-tips and summarizing policy documents.  
  
Key steps in loading the model:  
1. The tokenizer is loaded to process text into tokens that the model can understand.  
2. The model is loaded with data type specifications (`float16` for CUDA or `float32` for CPU).  
3. If a GPU is available, the model is mapped automatically to CUDA for faster inference.  
4. The tokenizer’s padding token is set to the end-of-sequence token if it is not already defined.

# Functions

## generate\_response(prompt, max\_length=1024)

This is the core function responsible for interacting with the Granite model. It accepts a prompt (user input or system-generated query) and generates a textual response. Key parameters include:  
  
- prompt: The input text to which the model will respond.  
- max\_length: The maximum length of the generated response (default: 1024 tokens).  
  
The function ensures GPU utilization if available and decodes the output into human-readable text. Temperature is set to 0.7 to balance creativity and relevance.

## extract\_text\_from\_pdf(pdf\_file)

This function extracts text from a PDF document. It uses PyPDF2’s `PdfReader` to iterate through each page and concatenate the extracted text. The function handles exceptions and returns an error message if the PDF cannot be read.  
  
Example usage:  
- Upload a PDF containing government sustainability policies.  
- The function extracts raw text which can then be summarized using the model.

## eco\_tips\_generator(problem\_keywords)

This function generates eco-friendly tips based on environmental problem keywords. It constructs a prompt using the user’s keywords (e.g., 'plastic waste', 'solar energy') and passes it to the `generate\_response` function. The model responds with practical, actionable tips.  
  
Example:  
Input: 'plastic pollution, recycling'  
Output: Suggestions on reducing plastic use, promoting recycling programs, and alternatives to single-use plastics.

## policy\_summarization(pdf\_file, policy\_text)

This function summarizes policy documents by analyzing either an uploaded PDF or directly provided text. The summary highlights:  
- Key points  
- Major provisions  
- Policy implications  
  
This feature is especially useful for policymakers, researchers, and citizens who need a quick understanding of lengthy policy documents.

# Gradio Interface

The Gradio interface provides a user-friendly way to interact with the system. It is divided into two main tabs:  
  
1. \*\*Eco Tips Generator\*\*  
 - Input: Environmental problem keywords.  
 - Output: AI-generated eco-friendly tips.  
  
2. \*\*Policy Summarization\*\*  
 - Input: Policy PDF file or direct text.  
 - Output: Concise summary including key provisions and implications.  
  
The interface uses Gradio components such as `Textbox`, `File`, `Button`, and `Tabs`. The `click` method binds buttons to their respective functions.

# Usage and Examples

To run the application, execute the script in a Python environment with the required dependencies installed. The Gradio app will launch locally and provide a shareable link.  
  
Example workflow:  
1. Launch the application.  
2. In the Eco Tips Generator tab, enter 'energy saving, water waste'.  
 - The system will output tips like using energy-efficient appliances and reducing water leakage.  
3. In the Policy Summarization tab, upload a PDF on renewable energy policy.  
 - The system will provide a summary with highlights of the main provisions.

# Real-world Applications

This tool has multiple applications in smart city planning and sustainability efforts:  
  
- \*\*Citizens\*\*: Gain practical eco-tips for sustainable living.  
- \*\*Government Officials\*\*: Summarize long policy documents for faster decision-making.  
- \*\*Researchers\*\*: Analyze policies and environmental trends.  
- \*\*NGOs and Activists\*\*: Spread awareness by converting technical policies into simplified summaries.

# Limitations and Future Work

While the application is powerful, there are some limitations:  
- The quality of summaries depends on the model’s training.  
- Some PDF documents may not extract text accurately (e.g., scanned PDFs).  
- Requires a stable internet connection to load the model from Hugging Face.  
  
Future improvements could include:  
- Support for multilingual summaries.  
- Integration with city dashboards for real-time policy updates.  
- Enhanced PDF processing for scanned documents using OCR.

# Conclusion

The Sustainable Smart City Assistance tool demonstrates how AI can be applied to address environmental challenges and improve policy accessibility. By combining eco-tip generation with policy summarization, the application empowers individuals and organizations to make informed, eco-friendly decisions. With further development, it can become a key component in building truly sustainable smart cities.