Project Documentation

Project Title: Citizen AI

Team Member: A.Tanya Snowlet

Abstract

The 'City Analysis & Citizen Services AI' project is designed to demonstrate how Large Language Models (LLMs) can support urban safety monitoring and provide government-related assistance to citizens. By integrating IBM Granite LLM with a Python and Gradio interface, the system delivers insights into crime indices, traffic safety statistics, and public service information. The project highlights the potential of AI in governance, transparency, and civic engagement.

1. Introduction

Cities are complex ecosystems that require efficient governance and citizen engagement. This project integrates AI-powered solutions to provide safety analysis of cities and act as a virtual assistant for government-related queries. By leveraging natural language processing, the system delivers easy-to-understand information to both citizens and officials.

2. Objectives

Provide real-time city safety insights such as crime index and accident rates.

Enable citizens to ask questions related to public services and receive AI-generated responses.

Offer a user-friendly interface for accessibility.

Demonstrate the integration of IBM Granite LLM with Python applications.

3. System Requirements

Hardware Requirements:

• Processor: Intel i5 or higher• RAM: 8 GB minimum• Storage: 2 GB free space• GPU (optional): CUDA-enabled GPU for faster processing

Software Requirements:

• Python 3.9 or higher• Libraries: transformers, torch, gradio• Internet connection for model download

4. Project Overview

Purpose: To analyze city safety and provide a government assistant tool.

Features:

City Analysis – Detailed reports on crime and accidents.

Citizen Interaction – Query-response system for civic issues.

Conversational Interface – Natural language support.

Gradio UI – Tabbed interface for smooth navigation.

5. Architecture

The architecture of the system consists of three primary layers:1. Frontend (Gradio): User interface with tabs for City Analysis and Citizen Services.2. Backend (Python + Torch): Handles model integration and function execution.3. LLM Integration (IBM Granite): Processes user input and generates human-like responses.The workflow begins with the user providing input (city name or query). The backend processes this input using the Granite LLM model, and the output is displayed on the Gradio interface.

6. Folder Structure

• app.py – Main script to launch the Gradio application.• model_loader.py – Handles Granite model loading and tokenizer setup.• utils/ – Utility functions for text processing.• docs/ – Documentation and related files.

7. Setup Instructions

Install Python 3.9 or above.

Install required libraries using pip install transformers torch gradio.

Download and load IBM Granite model.

Run the application script.

Access the interface via the provided local/hosted link.

8. Running the Application

Step 1: Launch the script in Python.Step 2: The Gradio app will create an interface with two tabs.Step 3: Enter a city name in the City Analysis tab to receive safety statistics.Step 4: Enter a query in the Citizen Services tab to receive policy information.

9. Functions / API Documentation

• generate_response(prompt, max_length): Generates AI-based responses.

• city_analysis(city_name): Provides safety analysis for the city.

• citizen_interaction(query): Responds to citizen queries.

10. Authentication & Security

Currently, the application is open for demonstration purposes. Future versions can include:• JWT or API key authentication• OAuth2 integration• Role-based access for administrators and citizens

11. User Interface

The interface is divided into two main tabs:1. City Analysis – Input city names and get crime/safety reports.2. Citizen Services – Input queries and receive government policy information.The UI is designed for simplicity, with textboxes and buttons for interactions.

12. Testing

Testing was conducted across multiple levels:

Unit Testing – Verified prompt and response functions.

API Testing – Checked response generation accuracy.

Manual Testing – Input of real and fictional cities.

Edge Case Testing – Handled invalid or empty inputs.

13. Known Issues & Limitations

Responses depend heavily on the underlying model and may vary.

No persistent database to store queries or responses.

Limited statistical accuracy as real-time datasets are not integrated.

14. Future Enhancements

Integration of real-time city datasets.

Enhanced visual dashboards with charts and graphs.

Mobile app version of the assistant.

Cloud deployment for scalability and multi-user access.

15. Conclusion

The 'City Analysis & Citizen Services AI' project showcases the application of AI in civic services. It provides valuable insights into city safety and offers assistance to citizens through a conversational interface. The project demonstrates the potential of AI-driven governance tools and paves the way for smarter, safer cities.