FastAPI Application for Text Processing Using OpenAI's GPT-4

-Kyush Maskey

Overview

This document provides a comprehensive summary of the FastAPI application developed for processing text by generating summaries and extracting named entities using OpenAI's GPT-4 model. The application also includes functionality to store and retrieve processed data from a SQLite database using SQLAlchemy.

1. Project Setup

1.1 Prerequisites

Before starting the project, ensure you have the following prerequisites installed:

- Python 3.7+: The application is built using Python, and it requires version 3.7 or higher.

-OpenAI API Key: An API key from OpenAI is necessary to access GPT-4 capabilities.

1.2 Cloning the Repository

To get started, clone the project repository to your local machine:

```bash
git clone https://github.com/edasaruhan/FTL_LLM_Maskey.git
```

1.3 Installing Dependencies

Install the required Python packages using the `requirements.txt` file:

```bash
pip install -r requirements.txt
```

1.4 Configuring Environment Variables

Create a `.env` file in the root directory of the project and add your OpenAI API key:

```env
OPENAI_API_KEY=your-openai-api-key
```

1.5 Initializing the Database

Before running the application, initialize the SQLite database by running the following command:

```bash
python -c "from main import init_db; init_db()"
```

2. Application Architecture

2.1 FastAPI Framework

The application is built using **FastAPI**, a modern, high-performance web framework for building APIs with Python 3.7+ based on standard Python type hints. FastAPI is chosen for its speed, ease of use, and the ability to quickly develop and deploy RESTful APIs.

2.2 Database Management with SQLAlchemy

SQLAlchemy is used as the Object-Relational Mapping (ORM) library for interacting with the SQLite database. The application stores the original text, its summary, and extracted named entities in a database table named `summaries`.

2.3 Environment Management with Decouple

The **decouple** library is used to manage environment variables, such as the OpenAI API key, ensuring that sensitive data is not hardcoded into the application.

2.4 Text Processing with OpenAI's GPT-4

The application interacts with **OpenAI's GPT-4** model to perform two primary tasks:

- **Text Summarization**: Generating concise summaries of the provided text.

- Entity Extraction: Identifying and extracting named entities such as people, organizations, and locations.

3. API Endpoints

The application exposes two primary endpoints:

3.1 POST `/process_text/`

- **Description**: Accepts input text, generates a summary, extracts named entities, and stores the results in the database.

- **Request Body**:

```json
{
  "text": "Your input text here."
}
```

- **Response**:

```json
```

```
{
  "summary": "Generated summary.",
  "entities": "Extracted entities.",
  "id": 1
}
```

Example of Process Text Link

3.2 GET `/summaries/{summary_id}`

- **Description**: Retrieves a stored summary and its entities by the given summary ID.
- **Response**:

```json
{
  "id": 1,
  "original_text": "Your input text here.",
  "summary": "Generated summary.",
  "entities": "Extracted entities."
}
```

Example of Summary Link

4. Usage Instructions

4.1 Starting the Application

To start the FastAPI server, run the following command:

```bash
uvicorn main:app --reload
```

This command starts the application in development mode with auto-reload enabled, making it easier to see changes during development.

4.2 Interacting with the API

- **Process Text**: Use the `/process_text/` endpoint to submit text for summarization and entity extraction.

- **Retrieve Summaries**: Use the `/summaries/{summary_id}` endpoint to retrieve a previously stored summary and its associated entities.

5. Conclusion

This application provides a robust foundation for text processing tasks using cutting-edge AI technology. By leveraging FastAPI, SQLAlchemy, and OpenAI's GPT-4, the application offers efficient and scalable solutions for summarizing text and extracting important information.

For further development, consider adding more endpoints, integrating additional models, or expanding the database schema to support more complex use cases.

6. References

- FastAPI Documentation
- SQLAlchemy Documentation
- OpenAI API Documentation