MenuData Chatbot Documentation

Project Overview

The **MenuData Chatbot** is a conversational AI designed to provide intelligent answers related to restaurants, their menus, and ingredients by utilizing both structured proprietary datasets and unstructured external sources. The project integrates **Retrieval-Augmented Generation (RAG)**, vector databases, and language models to ensure relevant, context-aware responses.

Key Objectives

- **Ingredient-Based Search**: Identify restaurants offering specific dishes.
- **Trending Insights**: Summarize emerging food trends.
- **Cultural & Historical Context**: Provide background on cuisines and dishes.
- **Comparative Analysis**: Compare menu prices across restaurant categories.
- **Menu Innovation Tracking**: Analyze ingredient usage trends over time.

System Requirements

Software & Libraries

- **Python**: Core development language.
- **LangChain**: Orchestration of retrieval-based LLM responses.
- **FAISS**: Vector search engine for efficient similarity retrieval.
- **Sentence Transformers**: Model for text embeddings.
- **Google Gemini API**: LLM used for chatbot responses.
- **Groq API**: Alternative LLM integration for fast processing.
- **Yelp API**: Fetches real-time restaurant data and reviews.
- **Wikipedia Library**: Retrieves historical and cultural insights.
- **Streamlit**: UI framework for interactive chatbot functionality.

Hardware Requirements

- **Local or Cloud Environment** with GPU acceleration for embeddings.
- **Sufficient Storage** for indexing structured datasets and embeddings.

Data Sources

The chatbot integrates data from multiple sources to enhance response accuracy.

1. Proprietary Restaurant Dataset (Structured Data)

The dataset contains structured information about restaurants and their menus:

- **Fields:**
  - restaurant_name, menu_category, menu_item, ingredient_name, menu_description, address, city, state, zip_code
- **Storage Format:** CSV files are loaded into **Pandas DataFrames** for processing.

2. Yelp API (Real-Time Data)

- **Purpose:** Fetches restaurant information, ratings, price levels, and customer reviews.
- **Integration:** The API is queried dynamically based on user inputs.
- **Not Embedded in FAISS:**
  - Since Yelp data **keeps changing** based on real-time reviews and restaurant updates, it is **not embedded into FAISS**.
  - Instead, it is fetched **dynamically** for each user query.

3. Wikipedia (Unstructured Data)

- **Purpose:** Provides historical and cultural insights about dishes, cuisines, and ingredients.
- **Retrieval Method:**
  - The chatbot **does not use the Wikipedia API directly** but instead **utilizes the wikipedia Python library** to fetch relevant summaries.
  - **Directly Embedded into FAISS**:
    - The fetched **Wikipedia summaries are stored in FAISS without further chunking**.
    - Since Wikipedia already provides **pre-summarized content**, additional chunking was deemed unnecessary in the current implementation.
    - However, **chunking can be implemented** to improve retrieval granularity.

4. News Articles (Planned but Not Included)

- **Purpose:** Track real-time culinary trends and emerging ingredients.
- **Reason for Exclusion:**

- Currently **not integrated due to performance limitations**.
- **News data would require chunking** before embedding to ensure efficient retrieval.

How to Run the Chatbot

1. **Open the folder "Code".**
   It has two files. One has the main code (Bot_Code.py); the other one has the code for running the UI (Bot_UI_Code.py).

2. **Run the Chatbot Locally**:
   Have the folder setup in your local environment and run the UI file using the following command.
   streamlit run Bot_UI_Code.py
   This will launch the chatbot in a web browser.

3. **Interacting with the Chatbot**:
   - Enter your query in the chat input.
   - The chatbot will retrieve relevant restaurant information, Wikipedia context, and Yelp reviews.
   - Responses are generated using Gemini LLM.

Technical Architecture

1. Data Ingestion & Preprocessing

- CSV files are loaded and structured data is extracted.
- Wikipedia summaries are fetched and embedded.
- Yelp API is used to gather real-time restaurant details.

2. Embedding & Vectorization

- **Model Used:** paraphrase-MiniLM-L6-v2 (Sentence Transformers)
- **Why This Model?**
  - **Efficiency**: Compact model with low computational cost.
  - **Speed**: Faster embeddings compared to larger transformer models.
  - **Effectiveness**: Sufficient for semantic similarity tasks in restaurant-related queries.

3. Vector Database for Search

- **Database Used: FAISS (Facebook AI Similarity Search)**
- **Why FAISS?**
  - **High-speed similarity search**: Optimized for large-scale vector retrieval.
  - **Low-memory footprint**: Performs well in local and cloud environments.

Current Limitations & Future Enhancements

1. Performance Constraints Due to Open-Source Models

- **LLM Performance**: Open-source models like Gemini-2.0 and paraphrase-MiniLM are **efficient but not the most powerful**.
- **Impact**: **Using larger, closed-source models (e.g., GPT-4 Turbo, Claude, or proprietary embeddings like OpenAI's Ada) would enhance speed and accuracy**.
- **Solution:** Future work could involve **hybrid models** that balance speed and computational cost.

2. Missing Reference Attribution

- **Issue**: References for retrieved content are not currently shown.
- **Reason**: The relevant code is **commented out** but can be re-enabled.

3. News Article Integration (Planned)

- **Current Status**: Not implemented due to **slow response times**.
- **Solution**: Pre-processing and caching articles would improve performance.

Execution Flow

1. **Query Processing**
   - Extracts **nouns & key terms** using **spaCy**.
   - Identifies **restaurant names, dishes, and locations**.
2. **Data Retrieval**
   - **FAISS search** finds relevant embeddings for restaurant and ingredient data.
   - **Wikipedia lookup** fetches historical context.
   - **Yelp API query** retrieves restaurant ratings, reviews, and prices.
3. **Response Generation**
   - Constructs a structured **prompt with retrieved data**.
   - Passes it to **Gemini LLM**, which generates the final response.

- **Response is formatted and returned to the user.**

Future Scope

- **Upgrade to More Powerful LLMs** (e.g., GPT-4 Turbo, Claude)
- **Real-Time News Tracking** (Optimized for faster retrieval)
- **Expanded Data Sources** (More restaurant databases, OpenTable API)
- **Advanced Price Analytics** (Direct menu pricing comparisons)
- **Multilingual Support** (Responses in different languages)