Adaptive Recommendation Chatbot with RAG and Vector Database

By: Anmol Valecha

July 2024

Introduction

Overview of the Project and Objectives

- The assignment involves developing a domain-specific chatbot application using a Large Language Model (LLM) for natural language understanding and a vector database (Annoy) for data storage and retrieval.
- **Objectives:**
  - Implement preprocessing of recipe data for a cooking and kitchen-related chatbot.
  - Build a backend using Flask integrating Annoy for similarity search, GPT-2 for text generation, and RAG for enhanced recommendation responses.
  - Create a user-friendly frontend with Streamlit for interacting with the recommendation system.

Virtual Environment Setup on Mac (M1 Chip)

Python Environment

1

- Install Python 3.9.7 via `pyenv`.
- Create a virtual environment (`venv`) and activate it.

Install required packages:

2

- `pandas`, `streamlit`, `requests` (frontend)
- `flask`, `sentence_transformers`, `transformers`, `annoy` (backend)

Model Downloads

3

- Download necessary models:
  - Sentence Transformers (e.g., `all-MiniLM-L6-v2`)
  - GPT-2 (for text generation)

Preprocessing Data

Data Cleaning and Embedding Generation

- **Data Cleaning**:
  - Remove duplicates and handle missing values in the "Healthy Indian Recipes" dataset.
  - Convert columns to appropriate data types (int, float) for analysis.
- **Embedding Generation**:
  - Utilize Sentence Transformers to generate embeddings from text data (Dish Name, Ingredients, Instructions).

Backend Implementation (backend.py)

Flask Backend Overview

- **Integration with Annoy**:
  - Load and query Annoy index for similarity search based on user input vectors.
- **GPT-2 Text Generation**:
  - Use GPT-2 via Hugging Face Transformers pipeline for generating detailed descriptions of recommended dishes.
- **Retrieval-Augmented Generation (RAG)**:
  - Implement RAG to enhance recommendation responses by leveraging pre-existing data stored in the system.

Frontend Implementation (frontend.py)

Streamlit Frontend Overview

- **User Interaction**:
  - Display a text input for users to query recipes.
- **Display Recommendations**:
  - Show recommended dishes ranked by relevance and distance.
  - Provide generated descriptions for each dish.

Commands to Run

Virtual Environment Setup on Mac (M1 Chip)

- **Install Python 3.9.7 via** pyenv**:**
  - pyenv install 3.9.7
- **Create and Activate Virtual Environment (**venv**):**
  - python3 -m venv venv
  - source venv/bin/activate

Package Installations

*Ensure you are in your virtual environment (*venv*) before running these commands.*

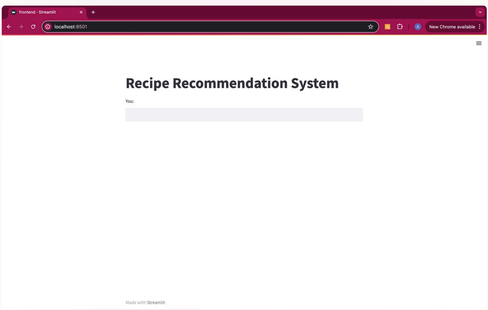
- **Backend Packages (**backend.py**)**
  - pip install flask
  - pip install sentence-transformers
  - pip install transformers
  - pip install annoy
- **Frontend Packages (**frontend.py**)**
  - pip install streamlit
  - pip install requests

Download Necessary Models

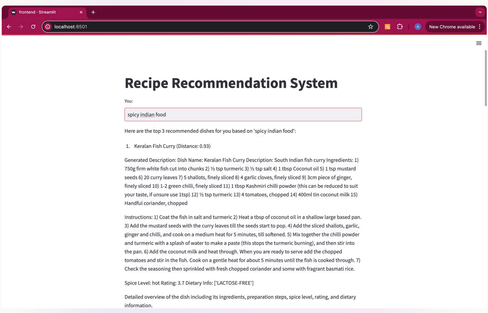
- Ensure to download the necessary models for Sentence Transformers and Transformers (GPT-2) as specified in your project setup. For example:
  - python -m transformers.cli.download_pretrained all-MiniLM-L6-v2

Examples of User Prompts

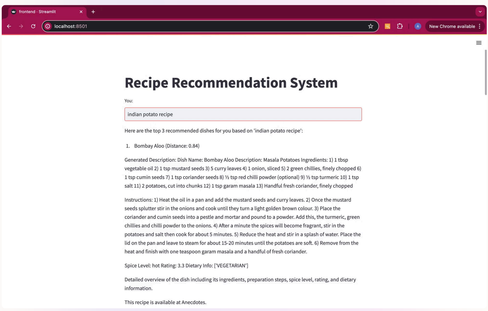
Bot Interface



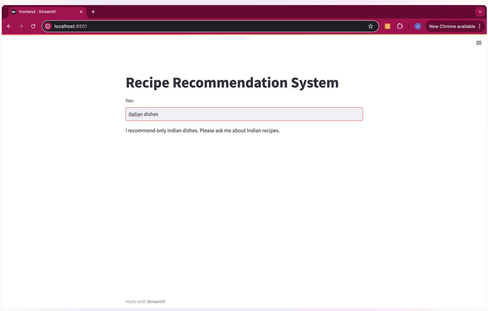
- **Example 1:** Asking for specific recommendations.
  - User Query: "spicy indian food"



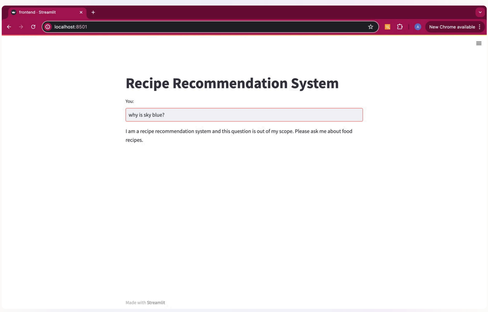
- **Example 2:** Inquiring about ingredient-based recipes.
  - User Query: "indian potato recipe"



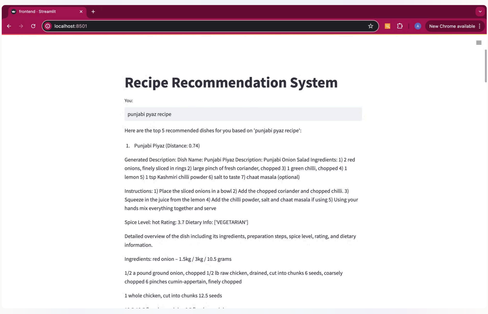
- **Example 3:** Asking for specific cuisine recommendations which is out of scope of dataset.
  - User Query: "italian dishes"



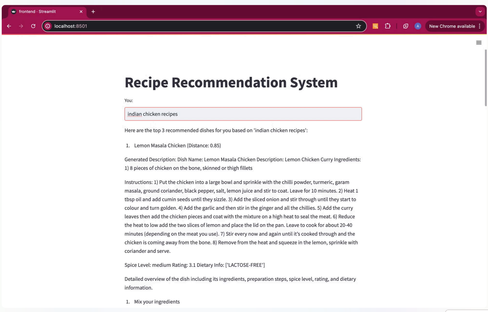
- **Example 4:** Testing the system's response to non-food-related queries
  - User Query: "why is sky blue"



- **Example 4:** Inquiring about specific recipes
  - User Query: "punjabi pyaaz recipe"



- **Example 4:** Inquiring about specific ingredient in recipes
  - User Query: "indian chicken recipe"



Refrences

- **GitHub Repository:**
  - Chatbot with RAG GitHub