Movie Mind Application

Marija Maneva

April 30, 2025

Contents

1.Introduction

1.1 App Description

The application is a movie recommendation system based on a conversational interface powered by a generative language model. Users can interact using natural language to receive personalized suggestions based on their tastes, past preferences or specific criteria such as genre, actors, directors or mood. The app provides detailed information about each movie, allows users to save favorites in a personal list and improves recommendations through the integration of a vector database. The result is an interactive and engaging experience that makes discovering new movies very simple and tailored to the user's needs.

2. Project Development

2.1 Setup

STEP 1: Create and Activate a Virtual Environment

Open the project folder and in the project root directory

```
# Create the virtual environment
python3.11 -m venv movie

# Activate the virtual environment
source movie/bin/activate #MacOS or Linux
Or
movie\Scripts\activate #Windows
```

STEP 2: Install Required Dependencies

```
# Core data processing and utilities
pip install pandas requests
pip install numpy==1.24.4

# Web interface
pip install gradio

# Environment variable management
pip install python-dotenv

# Vector database and embeddings
pip install chromadb sentence-transformers

# LLM integration
pip install langchain langchain-openai openai
```

Or just run the file requirements.txt

STEP 3: Set Up Environment Variables

Create a .env file in the project root directory to store API keys:

```
# Create .env file
touch .env  # On Windows, use: type nul > .env

# Add the following to your .env file:
OPENAI_API_KEY=your_openai_api_key_here
TMDB_API_KEY=your_tmdb_api_key_here
```

2.2 Data Collection

Go to directory src and run the provided movie_data_preparation.py script to download and prepare the data:

```
# Run the data preparation script
python movie_data_preparation.py
```

This script will:

- Download the MovieLens dataset
- Extract it to a 'data' directory
- Process the movie and ratings data
- Save the processed data to CSV files

Implementation of external APIs for data enrichment (retrievement of movie posters)

TMDB API

- Purpose: Retrieving movie poster URLs
- Implementation: Through custom TMDBHelper class

2.3 Implementation of the Vector Database

```
# Run the vector database script
python vector_database_setup.py
```

This script will:

1. create vector representations by using SentenceTransformer with 'all-MiniLM-L6-v2' model amd by converting text descriptions into vector embeddings;

2. implement ChromaDB as a vector database and store the embeddings locally in “data/embeddings” directory

2.4 Implementation of Recommendation System and Language Model

The MovieRecommender class is the core of the movie recommendation system. It uses ChromaDB to find movies that are similar in meaning to the user's request and OpenAI's language models through LangChain to create friendly, helpful responses.

The tools used in the script "recommendation_system.py" are :

LangChain

The framework creates conversation chains, manages prompts and handles context.
Components used:

- LLMChain - structures the conversation flow;
- PromptTemplate - formats inputs for the language model;
- ConversationBufferMemory - maintains chat history.

OpenAI API

The model creates conversational responses and personalized recommendations.
Implementation: through langchain_openai.OpenAI with temperature=0.7.

2.5 Creation of User Interface

The interface consists of an interactive web interface created with Gradio.

Run the interface script
python gradio interface.py

The script consists of:

- user interface framework - gradio and its various components for building the interactive web interface;
- data management - JSON for storing and loading user preferences;
- text processing - regular expressions for extracting movie information and posters;
- application logic - integration with MovieRecommender system;
- HTML/CSS - inline styling for custom visual elements;
- response processing - logic for transforming AI responses into user-friendly output.

Testing :
Unit tests were carried out for each script and they are in the folder called "tests".

To sum it up:

- Create virtual environment
- Install all the necessary dependencies
- Run movie_data_preparation.py script - to download the data
- Run vector_database_setup.py script - to create the vector database
- Run gradio_interface.py - to launch the app

3.Conclusions

The Movie Mind application presents a practical and functional solution for personalized movie recommendations through conversational AI. However, while the application does the core job, certain aspects, such as the management of favorite movies, remain too simplistic. At the moment, users can only store titles in a basic list without deeper interaction or filtering. Future development should focus on enhancing this feature to offer a more dynamic and user-friendly experience.