MovieMatch - Optimized with RAG: A Detailed Report

Introduction

MovieMatch has been revamped with the latest advancements in technology to enhance user experience and movie recommendation precision. By integrating Retrieval-Augmented Generation (RAG), a graph database, and an improved user interface, MovieMatch now offers a more powerful, intuitive, and efficient movie discovery experience.

Approach Taken

1. Integration with Retrieval-Augmented Generation (RAG)

Objective: Enhance the recommendation engine's capability to provide precise and contextually relevant movie suggestions.
Implementation:

- Leveraged OpenAI's models for generative capabilities, combined with Neo4j for retrieval tasks.
- Implemented a pipeline where user queries are processed to retrieve relevant data from the graph database before generating recommendations using OpenAI's models.

2. Graph Database Integration

Objective: Efficiently optimize movie search functionality.
Implementation:

- Chose Neo4j for its powerful graph database capabilities.
- Structured the database to maintain relationships between users, movies, and various attributes (genres, ratings, keywords).
- Used vector search techniques to enhance the search capabilities within the database, ensuring fast and relevant results.

3. Improved User Interface

Objective: Create a more intuitive and user-friendly design.
Implementation:

- Streamlined the UI to simplify navigation and enhance user engagement.
- Incorporated design elements that make the movie discovery process seamless and enjoyable.

4. Advanced Search Functionality

Objective: Enable detailed and flexible search options based on user preferences.
Implementation:

- Developed search features allowing users to filter movies by plot, ratings, and specific keywords.
- Implemented algorithms to handle complex search queries efficiently, improving the accuracy of recommendations.

Challenges Faced and Solutions

1. Data Integration and Management

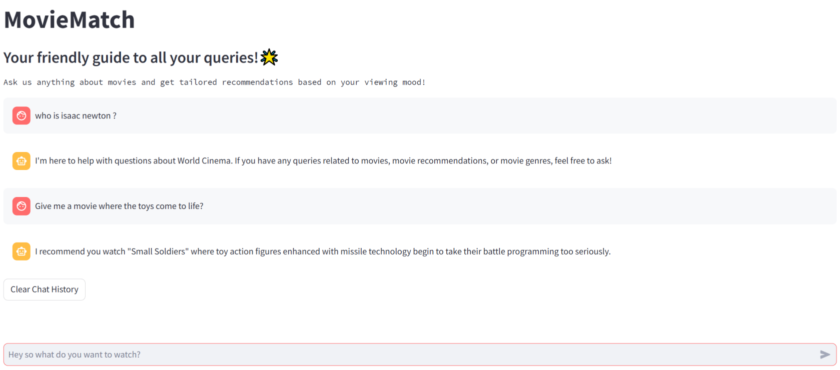
Challenge: Integrating diverse sources of movie data into a cohesive dataset.
Solution:

- Developed a custom dataset by collating data from various online sources.
- Used scripts to preprocess and standardize the data, ensuring consistency and compatibility with the graph database.

2. Modularising Code

Challenge: Ensuring the codes are written in a proper concise manner.
Solution:

- Used relative paths to import files from the subdirectories.
- Structured the project into subfolders, each containing specific functionalities.



MovieMatch

Your friendly guide to all your queries!

Ask us anything about movies and get tailored recommendations based on your viewing mood!

who is isaac newton ?

I'm here to help with questions about World Cinema. If you have any queries related to movies, movie recommendations, or movie genres, feel free to ask!

Give me a movie where the toys come to life?

I recommend you watch "Small Soldiers" where toy action figures enhanced with missile technology begin to take their battle programming too seriously.

Figure 1: Sample User Queries

3. Utilising Vector Searches

Challenge: Leveraging Vector embeddings offered by Neo4j.
Solution:

- Created Vector embeddings using GenAI functionality after rigorous testing.
- Integrated these embeddings into the Neo4j graph database to enhance the precision and relevance of search results.

4. Handling User Queries

Challenge: Ensuring the application doesn't veer away from the prompt.
Solution:

- Experimented with various prompt formulations to find the most effective ones.
- Refer to Fig 1.

Getting Started with MovieMatch

1. **Clone the Repository:**

   ```
   git clone https://github.com/your-repository/movie-match.git
   ```

2. **Install Neo4j Desktop and GenAI Stack:**
   - Follow Neo4j's GenAI Stack installation guide.
   - Request the dataset by emailing abhishekshankar79@yahoo.com.

3. **Set Up the Environment:**

```
python -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

4. **Configure Environment Variables:**

```
NEO4J_URI=*****************
NEO4J_USER=*****************
NEO4J_PASSWORD=*****************
NEO4J_DATABASE=*****************
OPENAI_API_KEY=*****************
OPENAI_BASE_URL=*****************
```

5. **Run the Application:**

```
streamlit run app.py
```

Conclusion

With these enhancements, MovieMatch stands out as a leading movie recommendation platform, offering users a seamless and personalized movie discovery experience. The integration of RAG, a graph database, and an advanced UI ensures that MovieMatch is not only user-friendly but also powerful in delivering accurate and engaging movie suggestions.

Enjoy exploring movies like never before with MovieMatch – your personal movie guide!

For more details, visit our Preview of Latest Updates.