A Retrieval-Augmented Film Recommendation System

Richard Anthony Álvarez

IPHS 484 Senior Seminar (Spring 2024) Prof Leibowitz, Kenyon College

Abstract

This project introduces a Retrieval-Augmented Film Recommendation System, developed as a Command Line Interface (CLI). It leverages advanced semantic search techniques, combined with retrieval-augmented generation, to deliver personalized movie recommendations. The system accesses extensive film metadata from two remote API sources, enriching the quality and accuracy of its suggestions. Tested across various movie preference profiles, the system adeptly adjusts its recommendations to suit individual tastes, showcasing its adaptability and the effectiveness of retrieval-augmented technology in streamlining user interactions with digital entertainment platforms.

Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) combines the capabilities of Large Language Models (LLMs) with the precision of information retrieval to enhance recommendation systems. Our film recommendation system partially adopts this approach by integrating a language model with a selective retrieval component. Rather than searching through an exhaustive database, our system strategically retrieves metadata from selected films based on initial user inputs, such as preferences for specific film genres or styles. This metadata, sourced from two external APIs, includes critical details like movie descriptions, ratings, genres, and user reviews.

The selected data then serves as contextual input for the language model, which is tasked with generating personalized recommendations. The language model, trained on a diverse range of movie-related datasets, synthesizes this information to grasp nuanced user preferences and suggest films that closely match the specified criteria. This targeted approach not only boosts the relevance of the recommendations but also ensures that the suggestions are insightful and contextually appropriate.

Currently, our system employs a one-shot prompting method with the language model, using metadata from specifically chosen films rather than aggregating data from a vast dataset. This adherence to a RAG philosophy lays the groundwork for future enhancements. As development progresses, the system could evolve to dynamically fetch and utilize broader data points from extensive databases in real-time, bridging the gap between traditional content-based filtering and the nuanced demands of personalized recommendation systems. This future potential makes our approach particularly suited for advancing in the dynamic field of digital entertainment.

Implementation

The Retrieval-Augmented Film Recommendation System was implemented using Node.js, a powerful and flexible runtime environment that excels in handling asynchronous operations and network requests, which are crucial for interacting with external APIs. The architecture of the CLI tool is designed to maximize the efficiency of data retrieval and processing through modular components, each responsible for distinct aspects of the recommendation pipeline.

API Integration: The system interacts with the OMDb and TMDb APIs to fetch detailed movie metadata. This is accomplished through bespoke modules (omdb.js and tmdb.js), which handle HTTP requests and responses. Node.js's fetch API is used to asynchronously retrieve data, ensuring the system remains responsive.

Data Processing: Once data is retrieved, it is processed and aggregated. The aggregateMovieData function merges data from both APIs, enhancing the richness of the movie profiles used for making recommendations.

User Interaction: User inputs are handled via the inquirer library, which facilitates CLI interactions. Users can input their movie preferences, which are then used to refine searches and tailor recommendations.

Recommendation Logic: The core recommendation logic is powered by LangChain's OpenAI integration. This involves generating descriptive analyses of user preferences and querying the TMDb API based on these insights. The system utilizes OpenAI's GPT models to dynamically generate queries that reflect user preferences, significantly improving the relevance of the movie suggestions.

This approach not only leverages Node.js's strengths in handling I/O-bound tasks but also integrates advanced AI and machine learning techniques to provide a highly responsive and intuitive user experience.

Example Film Recommendation

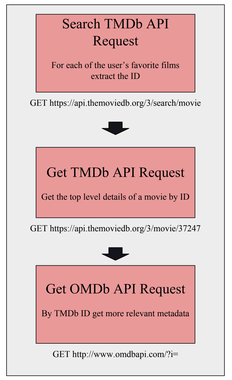
User Preference Description
Common genres: Drama, Romance, Comedy, Crime, Action, Thriller, Western, Mystery
Release date: 1967-2001
Runtime: 81-106 minutes\
Spoken languages: English, Spanish, Italian
Vote average: 6.7-7.6
Box office: $0-$33,600,000
Rating: PG-13, R

TMDb Discover API URL
https://api.themoviedb.org/3/discover/movie?include_adult=false&language=en-US&sort_by=vote_average.desc&primary_release_date.gte=1967-01-01&primary_release_date.lte=2001-12-31&vote_average.gte=6.7&vote_average.lte=7.6&with_genres=18|10749|35|80|28|53|9648|37&with_runtime.gte=81&with_runtime.lte=106&with_original_language=en|es|it&without_watch_providers=8|9|10&region=US

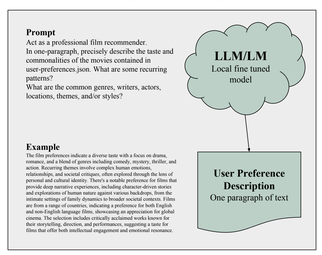
Suggested Film
- No Sympathy for the Devil
- The Hunted Lady
- Eyes of the Prey
- The Belle of Amherst
- The Walking Stick
- No Picnic
- Love God
- A Voyage Round My Father
- Billy Elliot
- Tres lancheros muy picudos
- Toy Story 2
- Ferris Bueller's Day Off
- Harold and Maude
- Man Facing Southeast
- Skylark
- Cria!

1. Ingest a list of favorite films

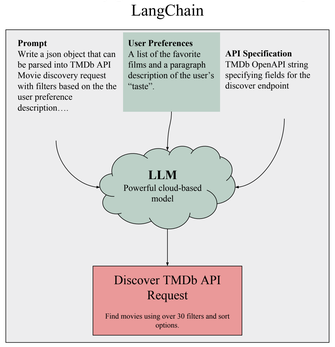
2. Create document of movie metadata



3. Create a description of user preferences



4. Discover movies by generating a tailored API request using LangChain



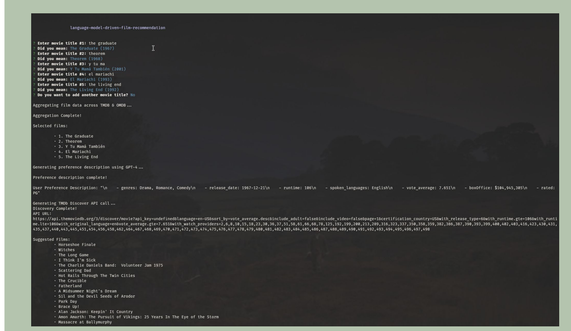
(Figure 1. Large Language Model-Based Film Recommendation System Architecture)

Conclusion

The Retrieval-Augmented Film Recommendation System has demonstrated its ability to provide novel and relevant insights into personalized film recommendations. While the system offers some transparency, the explainability of its processes and decisions could benefit from further testing and enhancement. This would improve user trust and satisfaction by making the recommendation logic more accessible and understandable.

Looking ahead, there are plans to evolve the system into a full Retrieval-Augmented Generation (RAG) model by establishing a proprietary database that indexes comprehensive film metadata. This will enable the system to perform more in-depth searches and generate richer user profile descriptions, leading to more accurate and personalized film suggestions.

The implementation of this complete RAG system will allow for more dynamic interactions with the data, significantly enhancing the system's capability to meet the nuanced preferences of its users.



(Figure 2. Demonstration of Application)

References

Alvarez, R. (2024). Retrieval-Augmented Film Recommendation System. GitHub Repository. Available at: https://github.com/raulduk3/language-model-driven-film-recommendation

OpenAI. (n.d.). OpenAI API. Available at: https://openai.com/api/

The Movie Database (TMDb). (n.d.). TMDb API Documentation. Available at: https://developers.themoviedb.org/3/getting-started/introduction

Internet Movie Database (IMDb). (n.d.). OMDb API. Available at: http://www.omdbapi.com/

LangChain. (n.d.). LangChain Library. Available at: https://langchain.com/

Node.js Foundation. (n.d.). Node.js Official Documentation. Available at: https://nodejs.org/en/docs/

Inquirer.js. (n.d.). Inquirer GitHub Repository. Available at: https://github.com/SBoudrias/Inquirer.js