Movie Recommendation System

Punnose Kozhuppakalam Thomas, Deepa Krishnan, Loshma Latha Babu,
Sukhveer Kaur, Princy Bindhu

January 12, 2025

1 Introduction

This project aims to create a Movie Recommendation System that helps users easily discover movies based on their personal preferences. With so many movies available on streaming platforms, it can be difficult for users to find content that they truly enjoy To solve this, the system uses Knowledge Graphs to connect movies, genre, actors and directors. By organizing this information, the system is able to suggest movies that match the user's tastes and interests. The system is also uses RDF and SPARQL to make sure the recommendations are as accurate as possible by understanding the connections between different movie details, the system provides personalized suggestion that are more relevant to the user. Ultimately the goal is to make more discovery easier and more enjoyable for users.

2 Application Description

2.1 Application System

The Movie Recommendation System comprises the following components.

1. **Front-End**

   The system has a user-friendly interface built using HTML, CSS and Java Script with pages for searching movies and viewing recommendations.

2. **Back-End**

   The system is developed using Python with with fast API to handle API requests, where controllers process user inputs, manage queries and return results.

3. **Knowledge Graph**

   The RDF database model relationship between movies, genres, actors and other attributes, while SPARQL queries retrieve relevant recommendations from this database.

4. **Data Flow**

   User input is processed in the front-end and back-end controllers query the RDF database to display the results as movie sugestions.

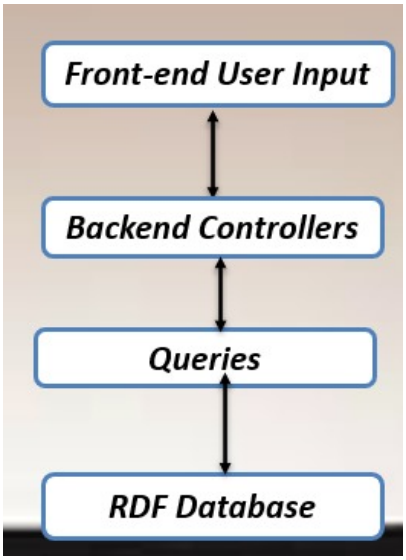

Figure 1: Interaction Flow Diagram

2.2 Technical Details

Data Source

The ontology was inspired by an older GitHub repository but was enhanced using three Kaggle datasets.

- The Movies Dataset by Rounakbanik(primary dataset).
- TMDB Movies Dataset (used for Year of release).
- IMDB Dataset of top 1000 Movies(used for Movie Poster URLs).

Ontology and RDF Conversion

- The ontology models defines relationships such as "directed_by" "acted_in," and "belongs_to_genre".
- Data from CSV files were converted into RDf Objects as per the reference schema using a Python script.

Tools and Libraries

- **Technologies Used:**
  - Knowledge Graphs and RDF for fdata modeling.
  - SPARQL for querying data.
  - Python for back-end development and data processing
  - Fast API for building APIs.
  - HTML, CSS and Java Script for creating the front-end user interface.

- **Libraries:**

rdflib	: for RDF handling.
uvicorn	: to run the back-end API endpoint instance.
fastapi	: to instantiate API endpoint instance.
tqdm	: to log progress when parsing/filtering.
kagglehub	: to load dataset from kaggle.
pydantic	: to specify the class patterns for expected input data in back-end API endpoints.
json	: for data processing.
os	: to get paths of files such as the RDF file.
logging	: to log progress to a seperate text file.
datetime	: to handle date and time data.
socket	: to get IP address dynamically for back-end.
pathlib	: to load paths for returning front-end HTMLfiles.

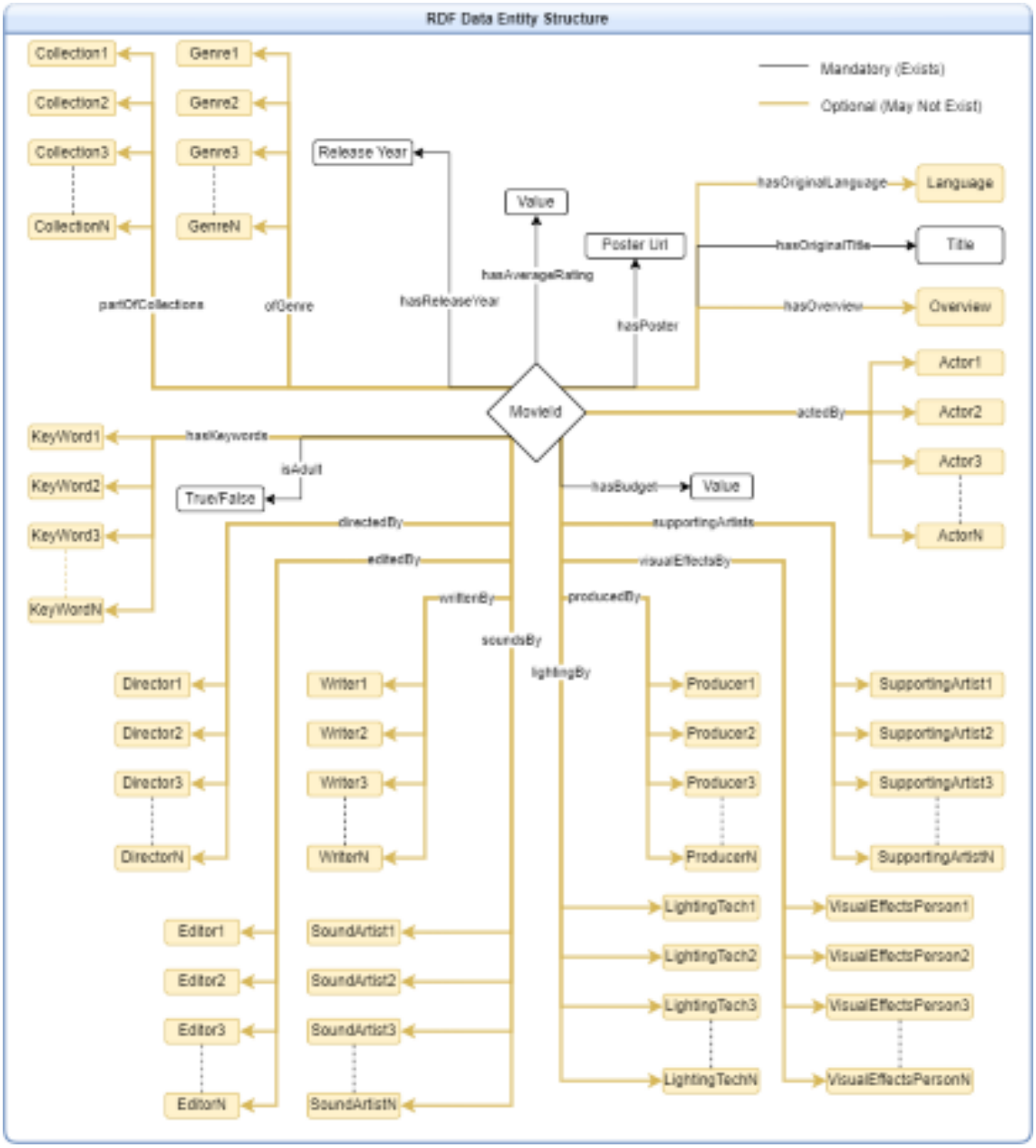

Figure 2: RDF Structure

2.3 Application Usage

Use Case 1: Movie Search
Scenario : A user searches for a specific movie.
functionality : The system retrieves movie details, including title, cast, genre and poster.

Use Case 2: Recommendations Based on Previous Movie Search
Scenario : A user views a movie's details.
Functionality : Recommendations are generated based on similar attributes such as the same genre, share

Use Case 3: Advanced Search
Scenario : The user specifies filters like year, actor or genre.
Functionality : The system applies these filters and displays matching movies.

The following screenshots provide visual overviews of the application's key interfaces: the index page, movie search functionality and recommendations of similar movie

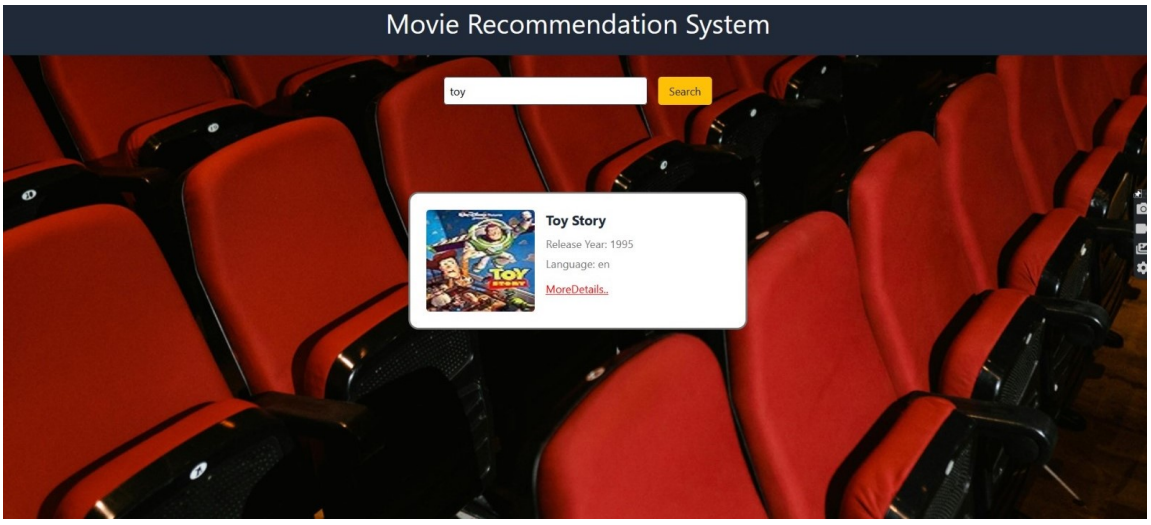

Figure 3: Index page

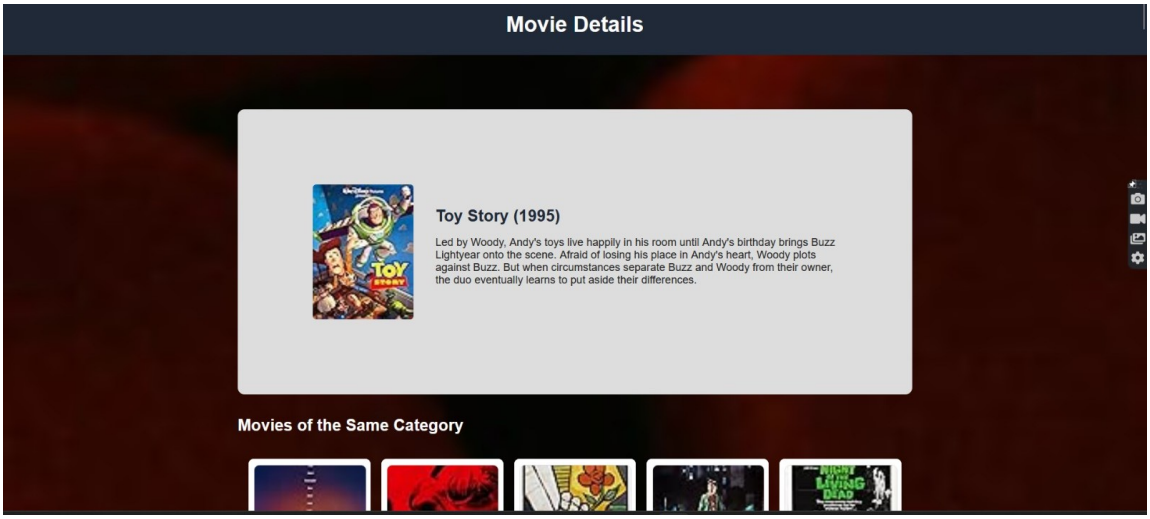

Figure 4: Movie Details

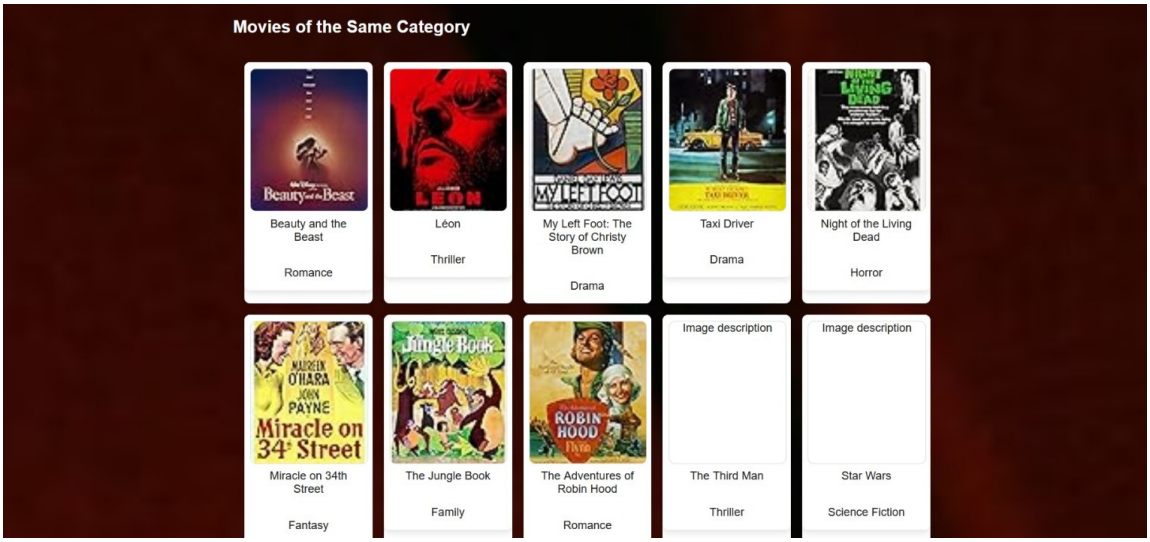

Figure 5: Recommendation of similar movies

3 Discussion

3.1 Challenges

- Handling Large RDF Datasets: The size of the RDF dataset and the complexity of SPARQL queries make query execution slower as the data grows.
- Dynamic Data Loading: Incorporating new movie data dynamically requires requires reloading and reindexing the RDF dataset.
- Query Optimization: Crafting SPARQL queries to ensure both accuracy and speed is challenging, especially for complex relationships.

3.2 Future Enhancements

- Incorporating User Feedback: Allow users to rate recommendations, enabling the system to refine suggestions based on feedback.
- Machine Learning Models Models: integrate AI/ML models to provide dynamic and behaviour-based recommendations.
- Real-Time Data Integration: Add live data feeds to ensure the latest movies trends are reflected in the recommendations.
- Cross-Platform Availability: Extend the application to mobile and desktop platforms for better accessibility.

3.3 Lessons Learned

- RDF and SPARQL: These technologies provide a robust way to manage semantic relationships but need careful optimization for scalability.
- Data Cleaning and Ontology Alignment: Combining multiple datasets into a unified ontology requires significant data cleaning and alignment efforts.
- FastAPI effieciency: FastAPI proves to be an efficient framework for backend API development, offering flexibility and performance.

4 Conclusion

The Movie Recommendation System demonstrates how Knowledge Graphs and RDF can be effectively used to model and query semantic relationships for personalized recommendations. By integrating structured data from reliable sources and leveraging SPARQL

queries, the system provides accurate and contextually relevant movie suggestions based on user preferences. The application offers a lighweight backend built using FastAPI, enabling seamless interaction between the front-end and RDF database. While challenges like query optimization and scalability persist, the project highlights the potential of Knowledge Graphs as a foundational technology for recommendation systems. This project provided valuable hands-on experience in RDF, SPARQL, and API development, showcasing how semantic data modeling can improve user experiences. Future enhancements, including machine learning integration and improved scalability, could elevate the system's performanace and usability, making it a competitive solution for modern recommendation needs.

References

[1] Guarino, Nicola, Daniel Oberle, and Steffen Stab. "What is an ontology?" *Handbook on ontologies* (2009): 1-17.

[2] Banik, Rounak. "The Movies Dataset." Kaggle, 2017. [online]. Available: https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset. [

[3] Shankhdhar, Harshit."IMBD Dataset of Top 1000 movies and TV shows "Kaggle,2021.[Online].Available: https://www.kaggle.com/datasets/harshitshankhdhar/imdb-dataset-of-top-1000-movies-and-tv-shows.

[4] Asaniczka. "TMDB Movies Dataset 2023 (930k Movies)." Kaggle,2023 [Online]. Available: https://www.kaggle.com/datasets/asaniczka/tmdb-movies-dataset-2023-930k-movies.