Knowledge and Data Engineer Project Proposal

Nikita Aksjonov, Nikos Kessidis, Bar Melinarskiy, and Konstantinos Zavantias

Introduction and Project Overview

Objective

This project focuses on designing and implementing a personalized movie recommendation system that seamlessly integrates Semantic Web technologies with the capabilities of a Large Language Model (LLM) to process both structured and unstructured data sources. By combining structured data from RDF-compatible datasets with embedding techniques for unstructured data, into a robust knowledge graph in Neo4j database to drive personalized recommendations [1].

In order to query the database containing the knowledge graph, to enhance usability, a web-based interface will allow users to seamlessly interact with the system, ensuring an intuitive and user-friendly experience for exploring personalized recommendations.

From the user interface (UI), the recommendation engine will offer various filters to query the data. Users can input their movie-watching history or preferences, such as genre, release year, language, dubbing options, or ratings, to refine their results. Once the query is processed in the application's backend, a list of the ten most relevant movie recommendations, based on affinity with the user's input, will be displayed in the UI.

Motivation

With the ever-growing number of entertainment options, users often struggle to find movies that suit their preferences. Recommendation systems play a vital role in addressing this issue, but many still focus solely on either structured or unstructured data, missing the potential of combining the two. This project aims to bridge this gap by integrating Semantic Web technologies and machine learning, enabling a deeper understanding of movie relationships and user preferences. By leveraging both data types, the system aspires to provide recommendations that are not only accurate but also finely tailored to individual tastes, enhancing the overall user experience.

Data Sources

Structured Datasets

In this project, structured datasets form the foundation for creating the semantic knowledge graph. These datasets leverage RDF (Resource Description Framework) standards to provide interconnected metadata about movies, actors, genres, and other relevant entities [2]. Below are the primary structured datasets selected for integration.

- **Wikidata:** A comprehensive dataset providing metadata on movies, actors, genres, and related entities [3].
- **DBpedia:** Structured knowledge extracted from Wikipedia, complementing Wikidata with additional semantic relationships [4].

These structured datasets serve as the foundation for constructing the semantic knowledge graph, providing rich, RDF-based metadata essential for creating a graph database. Wikidata offers comprehensive and well-structured information about movies, including genres, actors, and other related entities, making it an ideal source for defining nodes and relationships in the graph. DBpedia enhances this foundation by contributing additional semantic connections extracted from Wikipedia, adding depth and context to the graph. By integrating and combining these data sources, the project ensures a detailed and interconnected representation of the movie domain, enabling the development of a graph database optimized for personalized recommendations.

Unstructured Data Sources

Unstructured data sources provide supplementary information that is not readily available in RDF format. These datasets offer insights into user sentiments, contextual details, and nuanced relationships that enhance recommendation quality.

- **Wikipedia:** Articles containing detailed descriptions, synopses, and background information on movies.
- **IMDB:** User reviews and ratings, serving as a source for capturing sentiment and context via embeddings.

These unstructured data sources enrich the recommendation system by providing detailed and nuanced information that complements structured data. Wikipedia offers comprehensive textual content that can be processed into embeddings to enhance semantic understanding. IMDb contributes user-generated content, such as reviews and ratings, providing valuable insights into audience sentiment and preferences.

Preprocessing and Integration

To ensure consistency, structured datasets will undergo normalization and deduplication of entities. Unstructured data will be processed using embedding techniques, and the resulting vector representations will be linked to structured data entities using unique keys such as movie titles (URIs).

Proposed Application Features and Expected Outcomes

As shown in Fig. 1, our solution consists of five main components:

1) **Fetching Movie Data from RDF-Structured Datasets** Using SPARQL, we will query RDF-based datasets (e.g., Wikidata, DBpedia) to extract movie features such as titles, release dates, actors, producers, languages, genres,

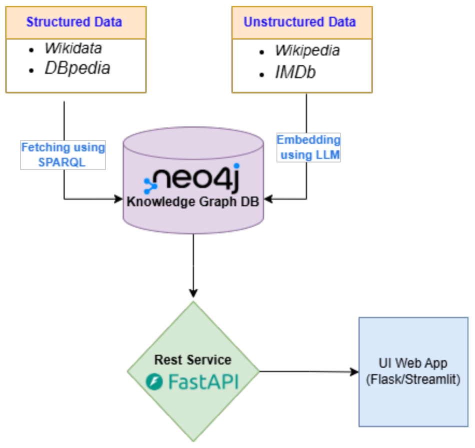
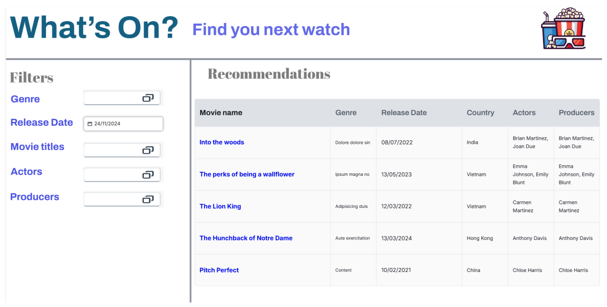


Fig. 1. Diagram of proposed solution.

and more.

2) **Fetching Movie Data from Unstructured Sources**
To incorporate free-text information (e.g., descriptions, reviews), we will use an LLM to generate RDF embeddings for unstructured data.
3) **Building a Knowledge Graph in Neo4j**
We will integrate all gathered data into a Neo4j knowledge graph, leveraging its built-in similarity functions (e.g., cosine, Jaccard) to support movie recommendations.
4) **RESTful Web Service**
Using FastAPI, we will implement a RESTful API to expose the database contents to the user interface in a simple and efficient manner. This approach will allow us to handle both straightforward queries—those that could be resolved using a single data source—and, more importantly, complex queries that require combining and integrating data from multiple sources.
5) **User Interface (UI)**
A user-friendly interface, as illustrated in Fig 2, developed with Flask or Streamlit, will allow users to:
   - Specify preferences via input filters.
   - View personalized movie recommendations based on their input.



Movie name	Genre	Release Date	Country	Actors	Producers
Into the woods	Dolore dolore sin	08/07/2022	India	Brian Martinez, Joan Due	Brian Martinez, Joan Due
The perks of being a wallflower	Ipsum magna no	13/05/2023	Vietnam	Emma Johnson, Emily Blunt	Emma Johnson, Emily Blunt
The Lion King	Adipisicing duis	12/03/2022	Vietnam	Carmen Martinez	Carmen Martinez
The Hunchback of Notre Dame	Aute exercitation	13/03/2024	Hong Kong	Anthony Davis	Anthony Davis
Pitch Perfect	Content	10/02/2021	China	Chloe Harris	Chloe Harris

Fig. 2. Mockup of UI for personalized movie recommendations.

Examples of queries we plan to support include the following, using the film *The Matrix* as an input:

- **Connected Films by Genre:** Suggestions such as *Inception*, *Tenet*, and *Interstellar* based on their shared sci-fi genre.
- **Films by the Same Director:** Movies directed by Christopher Nolan.
- **Films Featuring the Same Lead Actor:** Titles starring Keanu Reeves.
- **Films with similar plot** Recommendations such as *Dark City*.

Technical Implementation

Knowledge Graph Construction

Our proposed solution will have the following knowledge graph features (these are the main examples but will surely be extended at the end):

- **Node Types:** Movie, Actor, Genre, Director, etc.
- **Relationships:**
  - `ACTED_IN` (Actor → Movie).
  - `BELONGS_TO` (Movie → Genre).
  - `HAS_TOPIC_IN_PLOT` (Movie → topic).
- Embeddings derived from unstructured data will be stored as node attributes, enabling similarity computations.

Recommendation Algorithms

The recommendation engine will utilize a combination of:

- **Cosine Similarity:** Calculation of similarity between movie embeddings to rank recommendations.
- **Graph-Based Approaches:** Algorithms such as weighted shortest paths to account for graph relationships and user preferences.

Recommendations will be filtered based on constraints provided by the user, such as genre or release year.

Evaluation Metrics

The effectiveness of the system will be evaluated using the following metrics:

- **PrecisionK and RecallK:** Measuring the relevance of the top-K recommendations provided to users.
- **Node Similarity Scores:** Quantifying the semantic quality of relationships within the knowledge graph.
- **Resources Usage Performance:** Quantify the amount of running time, CPU, and memory usage of our selected solution.

I. References

[1] *Neo4j: The Graph Database Platform.* URL: https://neo4j.com/.

[2] Marcin Wylot et al. "RDF Data Storage and Query Processing Schemes: A Survey". In: *ACM Comput. Surv.* 51.4 (2018). DOI: 10.1145/3177850. URL: https://doi.org/10.1145/3177850.

[3] *Wikidata: A free and open knowledge base for humans and machines.* URL: https://www.wikidata.org/wiki/Wikidata:Main_Page.

[4] *DBpedia: Structured data extracted from Wikipedia for querying.* URL: https://www.dbpedia.org/.