IMDB Data Management and Movie Recommendation System

Richard Joerger
Rochester Institute of Technology
raj2348@g.rit.edu

Yifei Sun
Rochester Institute of Technology
ys8800@g.rit.edu

Ajeeta Khatri
Rochester Institute of Technology
ak6038@g.rit.edu

Zhuo Liu
Rochester Institute of Technology
zl9901@g.rit.edu

1. OVERVIEW

Our project is to build a Movie Recommendation System. The domain we chose is the IMDB dataset. The reason we choose it is not only because it contains a great amount of information about movies, like personal information of actors or actresses, but also a large number of film reviews and comments in it.

In general, our application will have two main functions. First is the basic function of our applications, to help users search for the movie they want by the filter they set. The second one is to generate a recommendation list for each user.

When it comes to movie searching, we will do this by using the Structured Query Language (SQL) to retrieve the relevant data from IMDB dataset to provide users with the movie details they want to see. The filter they set helps our application to define the query sentence. For instance, if one of our users is a crazy fan of Christopher Nolan, he can assign his name to the director name of the search filter, then the result will show them all Nolan's movies and all relevant movies.

About recommendation function, we will first generate training data set for each user since we have record of their search history and reviews. Finally we can build the machine learning model and get recommendation list for each user.

2. DATASET

The chosen IMDB dataset describes various aspects related to movies. Some of the examples of the data in the dataset is the data related to titles, actors, crews, episodes and ratings. All the data in the dataset is interrelated to different entities.

Below is a sample data representation. The data for actors can be represented by the following fields (nconst, primaryName, birthYear, deathYear, primaryProfession, knownForTitles) and a sample data is (nm0000001, Fred Astaire, 1899 ,1987, soundtrack, actor, miscellaneous, tt0043044, tt0050419, tt0053137, tt0072308). So, given this information Fred Astaire is the primary name of the person whose primaryProfession is soundtrack, actor, and miscellaneous. The fields for sample data for movie title can be represented as (titleId, ordering, title, region, language, types, attributes, isOriginalTitle). An example of this is (tt0000001, 1, Carmencita - spanyol tÃąnc,HU, ", imdbDisplay, ", 0). In this case, we know that this movie, Carmencita - spanyol tÃąnc, was released in Hungarian and is the original title. The attributes for the data related to episodes is as follows (tconst, parentTconst, seasonNumber, episodeNumber) and some sample data can be given by (tt0041951, tt0041038, 1, 9). This table is effectively a go between which helps maps an episode and its season to its entry in the title database. The dataset also contains the data regarding the movie crew, specifically the director and writer information for all the titles in IMDb where these fields are described (tconst, directors, writers). Sample data related to this can be represented as (tt0000001, nm0005690, "). This data is directly related to the principles as the tconst maps to values in the principles data. The data related to principles is given by (tconst, ordering, nconst, category, job, characters) and a sample data looks like (tt0000001, 1, nm1588970, self, ", ["Herself"]).The ratings for each movies is stored with the following attributes (tconst, averageRating, numVotes), where a single rating looks like (tt0000001, 5.6, 1533).

Some of the user scenarios for our systems could be querying for the movies titles based on the laguange of the movie. For example, if the user wants to find all the movies in English language, the query would look something like:

(Select * from title where language = 'en').

This would return a list of all the movies in our database with English language. In another scenario if the user wants to get a list of all famous movies casting his favourite actor, the user can also get the list of movies based on the actor. The query would looks something like.

(Select knownForTitles from actors where name = 'Brad Pitt').

These are some of the high level scenarios, we can further narrow them based on the filters selected by the user.

3. ESIMATED TIMELINE

Table 1 describes our expected timeline. The first week we expect to model our database using the ER model, convert it to its relational form , and then write and deploy the SQL script to create the tables on our MySQL database. Week 2 we will begin writing our database connector and a simple application which will use that connector to enter our data into the database such that we can begin working on the database with our real data. During week 3 and 4 we expect to write the UI portion of the application, at this moment we haven't determined what the UI will be. During the final week, week 5, we expect to just test our application and iron out any bugs found. The time line is intentionally very loose and intentionally has a large chunk of time dedicated to feature testing as this provides additional time in the event that we end up slipping past our projected timeline.

Table 1: Timeline

Week 1	ER Models, Conversion to Relational Model and creating MySQL tables	Done as a team
Week 2	Write DB connector and load data (split based on queries)	Zhuo and Richard: DB Connector. Ajeeta and Yifei: Data entry
Week 3	Write UI (split based on various components of UI)	Richard and Ajeeta: query view. Zhuo and Yifei: Data view
Week 4	Continue Writing UI (piece together UI)	Richard and Ajeeta: query view. Zhuo and Yifei: Data view
Week 5	Test(Verify that all goals are met)	Richard and Ajeeta: Test data view. Zhuo and Yifei: Test query view