**DATA MINING**

**INTRODUCTION:**

A recommendation system is used to recommend an item to a movie based on their preference. A recommendation system usually works either by finding similar users and recommend the items that are rated highly by the users similar to the user being recommended to or by finding the items that have similar features that a user might like based on the ratings given by the user to other items. The former strategy of recommending an item is called Collaborative approach while the latter one is called Content-based approach. Which strategy to choose mostly depends on what kind of information you have on users and on items. In this project we used content-based approach to build a movie recommendation system.

**DATA PREPARATION:**

We used **ml-latest-small** dataset from **movielens** for building the movie recommendation system. Files **ratings.csv**, **movies.csv**, **links.csv**, **genomes-tags.csv** and **genomes- scores.csv** were used. In addition, we used **name.basics.tsv**, **title.crew.tsv** from **IMDB** in order to recommend additional movies from the director of the movies a user has highly rated. These files were read into pandas DataFrame and then converted into appropriate form in order to use them to recommend movies to the users.

**SOLUTION APPROACH:**

Recommendation strategy of our Movie Recommendation system used Content Based Approach to recommend movies based on a user’s preference of movie features. In order to find out the features that a user may, the data files were read into different matrices which as follows:

**User-Behavior Matrix:**

It is a matrix that has user on its rows, movies on its columns and user’s rating as cell values. It shows ratings provided by user for movies. This matrix was built using data file ‘ratings.csv’ that had ratings given by each user to different movies. For simplicity, if any movie wasn’t rated by a user, then the rating given by the user to the movie was considered to be 0.

A screenshot of a cell phone

Description generated with very high confidence

Fig. User-Behavior Matrix

**Movie-Feature Matrix:**

The original Movie-Feature matrix has movies in row and columns in feature. We used total of 1149 features (1128 Tags given by users, 20 Genres and 1 released\_year) out of which. The cell values for each movie-feature were tag relevance (some number between 0 and 1) for tags given to the movie, 0 or 1 for genre depending on whether or not the movie belongs to the genre and numerical value of the year for released year. Data files ‘movies.csv’, ‘genomes-tags.csv’ and ‘genomes- scores.csv’ were used to build this matrix. I order to reduce the dimensionality of the matrix, we did Principal Component Analysis and used 585 principal components the sum of whose explained variance ratio was 95%. And thus we have a resulting matrix of reduce dimensions with movies in its row and principal components in column and PCA values in the cell.

A screenshot of a cell phone

Description generated with very high confidence

Fig. Movie-Feature Matrix

**User-Movie Matrix:**

Now in order to recommend movies to a user, we need to find out the user’s preference on movie feature and the movies with that feature. For this purpose, we took the dot product of the user behavior matrix with the movie feature matrix. The resulting matrix thus obtained would be user-feature matrix that basically gives us the idea of what feature each user looks for in the movies. Now taking the dot product of the resulting matrix and the transpose of the movie feature matrix gives us the user’s preference of each movie based on the feature the movie has. The resulting matrix thus obtained is the final user-movie matrix we use to recommend movies to the users.

A screenshot of a cell phone

Description generated with very high confidence

Fig. User-Movie Matrix

**Recommending movies to old users:**

From the user-movie matrix above, we now have an idea of each user’s preference on each movie. To recommend movie to a particular user, we now sort movies based on the user’s preference and recommend 25 movies the user haven’t already rated and has top preference values for the user.

**Recommending movies to new users:**

For new users, we need to first identify their taste. So, we first let them choose their five favorite genres. Then, they are showed the top-rated movies of those genres and allowed them to rate these movies. Now, as we have ratings for different movies from the user, we can recommend movies based on the strategy same as the one used to recommend to old users.

**Additional recommendation:**

The additional recommendation uses IMDB dataset to recommend movies to a user based on the director of the movies he is recommended using above strategy. This simply gets the name of the director of the recommended movies and recommend some other movies the director is famous for.

**INTERFACE:**

We have used Dash framework for implementing our front-end. It is user-friendly and easy to be used by both the old user and the new user. As Dash supports Single Page Application (SPA), instead of designing three different pages we have just designed three different layouts in a single page. The three different layouts are:

**User Login/Sign-Up:**

As the dataset provided by movielens had user id as the only information about user credentials, the system allows old user to login by their User ID. They enter their User ID in a text box just before “Sign In” button and then click the button.

For the new users, they can simply click on “New User” button to continue.

**Genre Selection:**

The Genre Selection layout is specifically designed for the new user to select their five favorite genres for recommending movies based on the selected genres. When they select the five genres, they can continue by clicking “Next” button. The “Next” button appears only when exactly five genres are selected which reduces the chance of getting undesired number of selected genres from user end.

**Movie Recommendation and Rating:**

This is the final and the most important layout of our application as it displays the recommended movies to users. In general, the layout displays 25 recommended movies and users can provide additional rating in a scale of 0-5 to provide more information regarding their preference which will later be used for making recommendation.

For old user, this layout shows recommended movies based on our recommendation strategy. Further, they can rate the movies that are recommended to refine the recommendation. Additionally, movies by the director of their top recommended movies are also displayed.

For new user, movies based on the picked genres are shown. As soon as they rate a movie, their User ID is created and stored in ratings.csv. They are now considered as old user. They can continue rating for refining recommendations.

Users can click refresh button on the top right, for a new set of recommended movies after they have rated one or more movies.

**EVALUATION:**

To evaluate the system, we cross-verified the features of the movies that was recommended to a user with the movies that the user has rated highly rather than performing evaluations that quantify how good or bad the recommender is. We verified the results for both the old and new users and from what we saw, the recommendation system mostly made appropriate recommendations. The verification for new users were done by highly rating the movies of a particular feature (based on genres and released year) and verifying that most of the recommended movies had the same features except for some movies that would get recommended to the user because of any other overlapping feature of the movies that the user rated high. But then again, this sticks with the idea of content-based recommendation since the system has no idea what feature made the user rate the movie high. Overall, the system was quite effective.

**CONCLUSION:**

The movie recommender is designed to recommend movies to both the existing users in the system as well as the users new to the system. The use of PCA, besides reducing the dimensionality helped us build a robust and efficient movie recommendation engine. Further improvements can be made to the system in order to handle larger datasets. Improvement in data preparation and pre-processing might speed of the execution.