Movie Recommendation System Project Report

1. Data Preprocessing & Feature Engineering

For this project I chose to use the MovieLens[1] dataset, which includes the following files:

Movies.csv contains 9,742 movies with their movieId, title, and genre; ratings.csv contains 100,836 rating records of 610 users on a scale of 1 to 5; Tag.csv contains 3,688 tag entries that describe the user's subjective impression of the movie.

The data cleansing process includes handling missing values and detecting anomalies. Missing types are filled in with "unknown," empty label entries are removed, and all ratings are verified to be within a valid range of 1 to 5.

Feature engineering involves encoding movie genres using the MultiLabelBinarizer, calculating average ratings and rating counts per user, extracting the year, month, and day of the week from timestamps, and applying TF-IDF to generate numerical features from the 50 most common labels.

2. Methods

2.1 Content-Based Filtering

The content filtering recommendation system analyzes the inherent characteristics of movies to recommend similar movies to users. In this project, the movie feature consists of the movie genre and the label. First, the movie types are uniquely thermally encoded using the MultiLabelBinarizer and the labels are TF-IDF encoded using the TfidfVectorizer. The following three indexes are used to calculate the similarity: cosine similarity, Pearson correlation coefficient and Jaccard index. Cosine similarity is achieved by cosine_similarity, which measures the Angle between movie vectors. The Pearson correlation coefficient is calculated by np.corrcoef to capture the linear relationship between features. The Jaccard index calculates the set similarity through the binarized features. The recommendation process is as follows: After selecting the target

movie, calculate its similarity with other movies, and recommend the top 5 movies in order of similarity from high to low.

2.2 Collaborative Filtering

Collaborative filtering makes recommendations based on user behavior data without the need for content characteristics. The project implements two collaborative filtering methods: user-based and item-based. User-based recommendation By calculating the cosine similarity matrix of the user rating vector, the most similar user groups are identified and recommended to the target users from the high score movies rated by these users. Object-based recommendations find other movies that are similar to the target movie by calculating the column similarity of the movie score matrix. The code logic is as follows: Create a user-movie scoring matrix using pivot_table. cosine_similarity calculates the similarity and sorts the first five recommended movies according to similarity and scoring values.

2.3 Matrix Factorization

The matrix decomposition method captures underlying interest patterns by decomposing the user-movie score matrix into implicit vector representations of users and movies. The Surprise library is used in the project to implement SVD and NMF. SVD(Singular Value decomposition) is trained by SVD() class with 50 potential factors and 20 training cycles; NMF(Non-negative Matrix factorization) is implemented by NMF(), ensuring that all factorization values are non-negative. During training, the data set was divided by 80:20 using train_test_split, and the model was cross-validated by cross_validate. Finally, the model performance was evaluated by accurate.rmse.

2.4 Deep Learning Recommendation Model (Neural Collaborative Filtering, NCF)

The project uses TensorFlow and Keras to implement the NCF model to merge content and collaborate information. The model architecture consists of two embedding layers, a fully connected layer and an output layer. The user and item embedding layers, both of 50 dimensions, are connected after flattening and input into two fully connected layers of 64 and 32 neurons, using the ReLU activation function to introduce nonlinear relationships. The output layer is a single neuron that predicts the score value. In the training process, the mean square error (MSE) is used as the loss function, Adam is used as the optimizer, and the learning rate is 0.001. The model was trained using 64 batches and 5

iterations, with a verification set ratio of 20%. After training, the prediction performance was evaluated using mean_squared_error and calculated Precision@K and Recall@K.

Similarity Computation

3.1 Cosine Similarity

Cosine similarity is measured by calculating the cosine value of the Angle between vectors, which is especially suitable for high and sparse data, so it has a good performance in the movie feature matrix. In the process of implementation, firstly, the type and label characteristics of movies are extracted from the data set, and the unique thermal coding process is carried out. Then, the cosine similarity between movie vectors is calculated based on the generated feature matrix, and the movie similarity matrix is obtained. Finally, according to the similarity matrix, the system will identify other movies that are closest to the movie features rated by the user as recommendation candidates.

3.2 Pearson Correlation Coefficient

The Pearson correlation coefficient measures the linear correlation between two variables and is often used in user-based collaborative filtering recommendations. In the implementation, it is necessary to calculate the similarity between different users based on the user scoring matrix to identify the user groups with similar scoring behaviors. The system then analyzes the rating patterns of these users, from which it can infer which movies the target audience is likely to be interested in. In order to avoid noise interference, the system filters out results with low similarity after calculation to improve the accuracy and reliability of recommendations.

3.3 Jaccard Index (Jaccard Index)

The Jaccard index is a set based similarity measure commonly used to measure the similarity of a set of movie labels. In the concrete implementation, the system first converts the label information of each movie into a set form and carries out binary representation. Then, the Jaccard similarity is obtained by calculating the ratio of intersection and union between the label sets. Through this method, movies with high overlap in label dimension can be effectively identified, and then

recommended content in line with users' interests and preferences can be generated.

Model Evaluation

4.1 Evaluation indicators

In this project, model performance was evaluated using the following metrics to measure the accuracy and ranking quality of recommendations.

RMSE (root mean square error) is used to measure the difference between the predicted score and the true score, and the smaller the value, the better the model predicts. In the code implementation, the mean_squared_error function is used to calculate the error and take the square root to get the RMSE.

Precision@K is used to evaluate the proportion of movies with a user rating of 4 or higher out of the first K recommendations. In the code, the accuracy of recommendations is measured by screening recommendations with a predicted score of ≥4 and calculating their proportion.

Recall@K measures the percentage of movies with user ratings ≥4 that are correctly recommended in the first K recommendations. In the process of implementation, the system counts the number of movies ≥4 in the real rating of users, calculates the proportion of recommended movies, and evaluates the recall ability of the recommendation system.

NDCG (Normalized Discount Cumulative Gain) is used to measure the ranking quality of recommendation lists. In the code, the ranking score of the predicted score is calculated, and compared with the best ranking, the normalized NDCG value is obtained to reflect the rationality of the recommended order.

4.2 Comparison of model performance

Recommendation models based on content filtering, collaborative filtering, SVD, NMF and NCF were trained and evaluated. For ease of comparison, matplotlib library was used to draw performance charts of each model on RMSE, Precision@K, Recall@K and NDCG.

References

[1]https://grouplens.org/datasets/movielens/32m/