Deep Learning - Based Movie Recommendation System

Javier Santamaría Pascual
Ingeniería Informática + CD & IA
17497745N

In this assignment, I will implement a movie recommendation system. In order to predict which movies can be more appropriate for a user, I have studied the possible approaches that we can reach with the given data.

The feature 'ratings' seemed particularly valuable, so the main objective of this deliverable will be to predict that column using the remaining features. Once a model with high prediction capacity is found, it can be used to estimate the ratings from a user for the films which has not yet seen. Films with higher ratings will be considered as the best possible recommendations for the user. For this reason, and to find the best model, different approaches will be used, and the one that reflects the best evaluation metrics will be lately used for the recommendation system.

As a common preprocessing step for all models, I loaded and stored the four available .csv files as DataFrames (movies, ratings, links and tags). Each model then required a slightly different preprocessing pipeline, which I describe below.

FIRST MODEL

In this first model, the objective will be to predict the rating using only data from the 100k dataset and, specifically, from ratings.csv and movies.csv. Firstly, some transformations have been applied: feature 'genres' as been transformed into dummy columns by using one-hot encoding so that each column will represent a different genre and the value in each column will be 1 if the movie belongs to that genre and 0 otherwise, I will merge both files by the column 'movieId', remove the columns 'title' and 'timestamp' and transforming all the features in numeric format for tensor compatibility. Secondly, the data will be split into training, validation and test sets (80-10-10 split). Thirdly, six new columns that are related to the ratings ('std_movie_rating', 'avg_movie_rating', 'count_movie_rating', 'avg_user_rating', 'count_user_rating', 'std_user_rating') have been created. It is very important to create those columns after the split because those metrics cannot be calculated using data from the testing set because we would be introducing leakage data and the predictions would not be real. The only data available to create those metrics is the one from the training set.

Finally, data is normalized using MinMaxScaler and all the rows with Nas are removed. Data is transformed into tensors, then into TensorDatasets and finally into DataLoaders(with a batch size of 1024). DataLoaders are the required structure to incorporate the data into the neural network. The shape of the input tensor X consists of 28 features per sample, and the output tensor y is a single scalar representing the normalized user rating.

A simple fully-connected neural network with layers: $28 \rightarrow 256 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 1$ including dropout(0.2) in two layers is trained and validated for 50 epochs using custom functions. This functions train the model with each batch of the DataLoader, produces predictions and the loss between predicted and actual values is computed using the defined loss function. After calculating the gradients via backpropagation, the optimizer updates the model weights and clears the gradient buffers to prepare for the next iteration. Validation function sets the model to evaluation mode to deactivate dropout layers and avoid unnecessary gradient computations. Both functions, on each iteration of the loop, display some metrics to compare the performance across epochs.

After the process, test is used to display some metrics about the performance of the model:

Metric	**Value**
R^2 Score	0.4452
MAE (Mean Absolute Error)	0.1217
MSE (Mean Squared Error)	0.0263
RMSE (Root Mean Squared Error)	0.1621
Precision	0.6199
Recall	0.9309

SECOND MODEL

This model adds data from the TMDB (the extraction of the data has been performed using a function that can be found in 'Get data from APIs.ipynb' in the following repository https://github.com/javiers2004/Movie-Recommendation-System. After the extraction, the next step is grouping all the data so a merge is required between ratings and links by the column 'movieId' and between links and the extracted data by the column 'tmdb'.

The preprocessing of the part of ratings is the same as in the previous model. However, it is important to make some transformations in the data extracted from the API. The title of the movie is removed, all the numeric data is normalized, only the year from 'release_date' is taken and the 'original_language' column is transformed by using one-hot encoding. As in the first model, the split into train, validate and test is done (60-20-20) and the DataLoaders are obtained (1024 of batch size). The features of this second model are the following ones: 'ratingTmdb', 'release_date', 'original_language', 'origin_country', 'votes,budget', 'revenue' and 'runtime'.

This architecture of this neural network will be a little different because the layer size for the input tensor is not 28 but 64 (it has many more features from TMDB): 64 $\rightarrow$ 256 $\rightarrow$ 128 $\rightarrow$ 64 $\rightarrow$ 32 $\rightarrow$ 1, with the same dropout layers. The train and validation functions are the same and the number of epochs has been increased to 100. As this models are not very big, the process of training and validating for 100 epochs does not take a significant amount of time.

This are the evaluating metrics of the model:

Metric	**Value**
R^2 Score	0.4699
MAE (Mean Absolute Error)	0.1182
MSE (Mean Squared Error)	0.0252
RMSE (Root Mean Squared Error)	0.1586
Precision	0.6185
Recall	0.9331

As it is evident, with this incorporation of data from the API, metrics have improved (R^2 and recall have increased while MAE, MSE and RMSE have decreased). So we can conclude that this second model is better than the previous one in predicting the ratings.

THIRD MODEL

This third model is like the second but instead of using the data from the TMDB API, it uses the IMDB API. The extraction function is also in 'Get data from APIs.ipynb' in the same repository. The preprocessing of the data between links and the extracted data is almost the same but the performed merge is by the column 'imdbId'. The features used in this model are: 'imdbId', 'BoxOffice', 'Country', 'Language', 'Metascore', 'Released', 'Runtime', 'imdbRating' and 'imdbVotes'. Features 'Country' and 'Language' are transformed into dummy columns using one-hot encoding. Columns are then normalized, split into train, validate and test (60-20-20) and converted into DataLoaders (1024 of batch size).

This third neural network is different from the previous ones because as the input tensor is quite bigger(1093 columns). The size of the network has been increased so the dimensions are: 1092 $\rightarrow$ 2048 $\rightarrow$ 512 $\rightarrow$ 256 $\rightarrow$ 64 $\rightarrow$ 16 $\rightarrow$ 1 , adding 3 dropout layers of 0.2.

By training the model during 50 epochs, the evaluation metrics obtained are:

Metric	Value
R^2 Score	0.4805
MAE (Mean Absolute Error)	0.1186
MSE (Mean Squared Error)	0.0244
RMSE (Root Mean Squared Error)	0.1563
Precision	0.6205
Recall	0.9401

Comparing the results of Model 2 (TMDB data) and Model 3 (IMDB data), we can conclude that the IMDB-based model provides slightly better overall predictive performance. It achieves higher values in R^2, Precision, and lower error metrics (MAE, MSE, RMSE), indicating more accurate rating predictions. Model 2 shows a slightly higher recall. However, the overall improvement in regression and ranking metrics suggests that the information obtained from IMDB is more informative for this recommendation task than the data from TMDB.

FOURTH MODEL

This model combines the second and the third models. So all the features used are: 'ratingTmdb', 'release_date', 'original_language', 'origin_country', 'votes,budget', 'revenue', 'runtime', 'imdbId', 'BoxOffice', 'Country', 'Language', 'Metascore', 'Released', 'Runtime', 'imdbRating' and 'imdbVotes'. The preprocessing step is the same and the only difference is the step of merging all them into only one dataframe. The split into training validation and test sets is the same. Normalization and transformation into DataLoaders are also computed.

The neural network is bigger because the input tensor requires bigger first layers but the structure of the network is similar too the last two previous models, with the same dimensions in the layers and the same dropouts.

The same training and validation functions are applied for 50 epochs and the obtained results are:

Metric	Value
R^2 Score	0.4629
MAE (Mean Absolute Error)	0.1223
MSE (Mean Squared Error)	0.0253
RMSE (Root Mean Squared Error)	0.1589
Precision	0.6534
Recall	0.9150

Model 4 does not significantly improve performance compared to using IMDB data alone (Model 3). In fact, Model 3 consistently outperforms Model 4 across nearly all key metrics: R^2, MAE, MSE, RMSE, and Precision.

FIFTH MODEL

Until this point, the data that has been used for ratings has consisted of 100k ratings from different users. In this final model, we utilize the 1M ratings dataset to explore whether increasing the amount of training data leads to better performance and more robust predictions. The only change with respect to the Model 4 is the step of reading the files (in this case is a .dat so just change the line of reading).

The features used, the steps of preprocessing, spliting the data, and training and validating are the same than in the Model 4. Due to the larger dataset, Model 5 required more time for training compared to the previous models. After 100 epochs, the results obtained are:

Metric	Value
R^2 Score	0.5611
MAE (Mean Absolute Error)	0.1054
MSE (Mean Squared Error)	0.0206
RMSE (Root Mean Squared Error)	0.1436
Precision	0.6502
Recall	0.9453

CONCLUSIONS

After evaluating each one of the 5 models this is a brief summary of the different metrics obtained:

	R^2	MAE	MSE	RMSE	Accuracy	Recall
Model 1	0.4452	0.1217	0.0263	0.1621	0.6199	0.9309
Model 2	0.4699	0.1182	0.0252	0.1586	0.6185	0.9331
Model 3	0.4805	0.1186	0.0244	0.1563	0.6205	0.9401
Model 4	0.4629	0.1223	0.0253	0.1589	0.6534	0.9150
Model 5	0.5611	0.1054	0.0206	0.1436	0.6502	0.9453

And these are some graphical representations of the data that will make it easier to draw conclusions about the models and identify which one is the best:

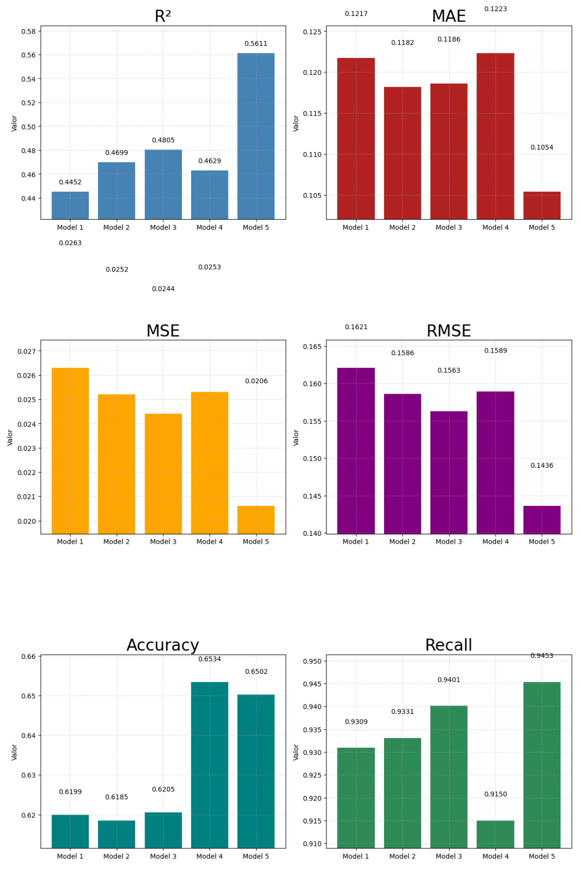

Figure 1: Histograms of the metrics per model

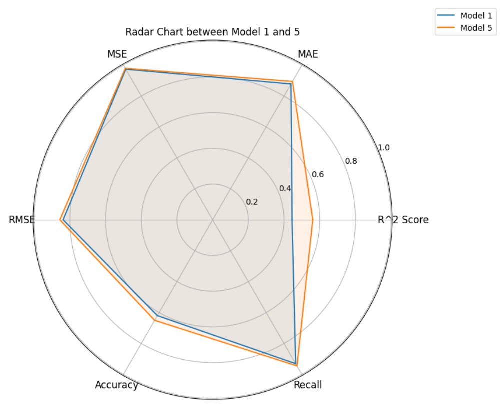

Figure 2: Radar chart between models 1 and 5

Analyzing the results we can draw the following conclusions:

1. **Performance Improvement with More Data:** The results show a clear trend where the addition of more data, particularly moving from Model 4 (using 100k ratings) to Model 5 (using 1M ratings) which use the exact same features, consistently improves certain metrics, especially R^2 score. Model 5 achieves the highest R^2 score of 0.5611, indicating that it explains more than 50% of the variance in the ratings. This suggests that using a larger dataset helps the model capture more complex patterns in the data.

2. **Error Metrics:** As the number of ratings increases, error metrics such as MAE (Mean Absolute Error), MSE (Mean Squared Error), and RMSE (Root Mean Squared Error) generally decrease. For instance, Model 1 has the highest MSE (0.0263) and RMSE (0.1621), while Model 5 has the lowest values at 0.0206 (MSE) and 0.1436 (RMSE). This indicates that larger datasets lead to more accurate predictions, reducing the discrepancy between predicted and actual ratings.

3. **Precision and Recall:** While precision fluctuates across models, it is relatively stable. Model 4 has the highest precision (0.6534), suggesting

it is the most effective at minimizing false positives compared to the other models. On the other hand, recall increases with each model, with Model 5 achieving the highest recall score of 0.9453, indicating that it successfully identifies most of the true positive cases. This suggests that while Model 5 is more precise, it also performs very well in identifying relevant recommendations, making it a balanced model for predicting ratings.

4. **NDCG@10:** NDCG@10 (Normalized Discounted Cumulative Gain at rank 10) remains perfect across all models (1.0000). This indicates that, in terms of top-10 recommendations, each model is performing equally well, regardless of the number of ratings or changes in the model architecture. This metric is not represented in the graphical visualizations but is calculated in the metric evaluation section on each model.

Overall Conclusion: Increasing the dataset size from 100k ratings to 1M ratings significantly improves the model's accuracy, with Model 5 being the best overall model. It strikes a good balance between prediction accuracy, precision, and recall. Therefore, we can conclude that using a larger and more diverse dataset can improve the quality of movie recommendation systems, especially when coupled with robust preprocessing and model architectures.

Finally, and after concluding that the Model 5 is the best one in predicting the rating that users would give to movies, this model will be used for the recommendation system so there is a function at the end of the notebook that, introducing the Id of a user, calculates the expected rating of each movie that has not been seen and the top 10 rated movies are displayed as recommendations to the user.

As we have discovered that the addition of more data improves the metrics of the model,a bigger ratings file could be used to get better recommendations for the users. More features such as actors or main argument of the movie could be used also to improve this results.

Finally, this research has underscored the importance of large, diverse datasets in training deep learning models, as they allow the model to learn more complex patterns and provide more robust predictions.