CS-423 DIS - Project 2 Report

Team `group1` — Aybars Yazıcı, Bojan Lazarevski, Can Kirimca
Department of Computer Science, EPFL, Switzerland
Kaggle notebook: (RMSE Score: 0.8)

I. Introduction

This report details the development of a recommender system for movie ratings, using the MovieLens dataset. Our project's goal is to accurately predict user preferences and suggest relevant content, utilizing a variety of data-driven techniques. The approaches discussed in this paper are as follows:

A. Matrix Factorization III-A
B. Collaborative Filtering III-B
C. SLIM III-C
D. Neural Network based approach III-D

II. Data Scraping and Analysis

The available data for this competition consisted of the following files:

- train_ratings.csv, a csv file containing a rating given to a movie by a user.
- tags.csv, movies.csv, links.csv files containing extra information about movies
- test_set_no_ratings.csv the test set to run our model on.

After a preliminary data analysis, one can see that the data does not contain a lot of extra information about the movies. The information on tags.csv when grouped by movieId and joined by movie id on our train set, reveals that more than 50% of the movies in the train set do not contain any tags. On top of this, the movies.csv only contains the title of the movie and its genres. Though all of our approaches can work with just a user-movie interaction matrix(i.e our train_ratings.csv file), our last approach will most definitely benefit from more features.

Considering these limitations, we decided to leverage the `links.csv` file, which contains links to IMDB and TMDB websites for each movie. Utilizing the TMDB API, we enriched our dataset significantly, scraping detailed movie information such as budget, revenue, runtime, vote average, vote count, and textual overviews. This enriched dataset, represented in `movies_df.xlsx`, thus became a cornerstone for our enhanced feature set.

Our further data analysis, detailed in the `data_analysis.ipynb` notebook, focused on extracting meaningful insights from this enriched data. For both the numerical and categorical features, we tried obtaining the Pearson Correlation and Mutual information to understand their relationship with user ratings (Figure 1). Again, for both of these feature types, we have also trained simple RandomForestRegressor and GradientBoostingRegressor, to assess their importance in predicting user ratings(Figure 2). The results obtained from these simple models were: for numerical features: 0.96 RMSE, meanwhile for genres 0.9977, and for tags 1.0256 (since we had a lot of unique tags, we only considered the top 20 most common ones for the model)

Apart from these numerical and categorical features, using the API we have also obtained the synopsis/overview of each movie. To be able to use this as a feature in our recommender system, we have used a general purpose sentence transformer model('all-mpnet-base-v2') to get the embeddings for each movie. But since these embeddings were really high dimensional (700+), we have then applied

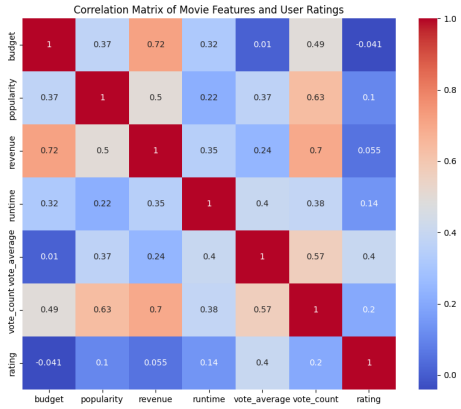

Fig. 1: Pearson correlation of our numerical features.

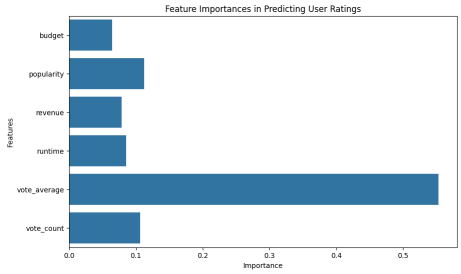

Fig. 2: Feature importance of the numerical features according to RandomForestRegressor

UMAP dimensionality reduction technique.. The implementation of this embeddings feature can be found in the following notebooks: `generate_embeddings.ipynb` and `embeddings_dim_reduction.ipynb`.

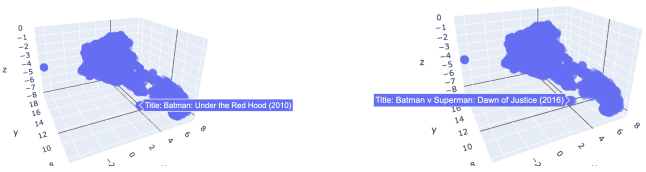

Fig. 3: The overview embeddings of movies reduced to 3D

Now that we have established a deep understanding on our data set, we move onto our methodologies(III).

III. Methodologies

A. Matrix Factorization

Our first approach involved matrix factorization, implemented in `first_approach.ipynb`. We optimized user and item feature matrices (P and Q) to predict ratings. The algorithm iteratively updated these matrices using gradient descent, with regularization to avoid overfitting. Thus at each step we sample one input data, and update the corresponding

movie and user vectors:

$$p_{ik} := p_{ik} - \alpha \left(\frac{\partial e_{ij}^2}{\partial p_{ik}} \right) = p_{ik} + 2\alpha \left(e_{ij} q_{kj} - \lambda p_{ik} \right)$$

$$q_{kj} := q_{kj} - \alpha \left(\frac{\partial e_{ij}^2}{\partial q_{kj}} \right) = q_{kj} + 2\alpha \left(e_{ij} p_{ik} - \lambda q_{kj} \right)$$

We also employed GPU-accelerated batch processing for efficient computation. This approach gave us an RMSE of 0.983.

We then also tried an SVD approach, implemented in `svd_factorization_custom.ipynb` we decompose our user-rating matrix into U, σ, V_t then predict the rating of user i for item j as the dot product of row i of U and column j V_t, we then add the global bias, user and item variance from the global bias to this prediction. This approach gave us an RMSE of: 0.935

We then also tried the surprise library to see how well our custom implementation fared. This can be found in `svd_factorization_surprise.ipynb`. The surprise library got an RMSE of: 0.8779

B. Collaborative Filtering

Our next approach is one of the most simple but still effective methods when dealing with recommender systems - collaborative filtering implemented in `collaborative.ipynb`. It makes predictions about a user's preferences or interests by leveraging information from the preferences and behaviors of a group of users. Firstly, we create the user-item pivot train and validation matrices where we store the ratings of each user for each movie. This is followed by the computation of the global mean rating as well as the user and item variance from the global average. We then calculate the user-user and item-item similarity matrices. Then to find the prediction for user i and item j, in user based CF we find the most similar k users, then take the weighted average of the ratings of item j of those users. This approach gave us an RMSE of 1.79. In item based CF, we find the most similar k items rated by our user i and then take the weighted average of them. This approach gave us an RMSE of 1.39.

C. SLIM

We then experimented with another approach, where we use a sparse linear method implemented in `slim_elasticnet.ipynb`. Particularly, we endeavored to construct the SLIM algorithm using ElasticNet regression. We trained the ElasticNet regression model to minimize the loss associated with SLIM and calculate the weight matrix. Our approach involved estimating movie ratings by calculating a linear combination of a user's other ratings using the corresponding entry of the weight matrix generated through ElasticNet. However, this process resulted in an excessively sparse weight matrix, consequently leading to lower predictions and a high RMSE around 2.4. Since this score is subpar, we don't include it in our results.

D. Neural Network with PyTorch

Implemented in `pyTorch.ipynb`, our neural network model was developed using PyTorch. We started with a simple approach, where we had 2 embedding layers, one of size $(N, embed\_dim)$ and one of size $(M, embed\_dim)$ where N=number of unique users and M=number of unique movies. After these embedding layers, we have a 2 hidden fully connected layers with ReLU activation function along with a dropout(of 0.2). Then a final layer to get the predictions. Which got us a test RMSE of 0.882

After this first approach, we then embarked on an approach that integrated only the most crucial features identified in II. This included user embeddings, genres, popularity, vote average, revenue, and dimensionally reduced overview embeddings (15 dimensions). In this iteration, we opted to exclude movie embeddings to mitigate overfitting, since we noticed training the model with our new features along with movie embedding resulted in an extremely low training error and a high test error. After removing the movie embeddings this approach resulted in an RMSE of 0.811.

However, leveraging the inherent capability of neural networks to discern and assimilate important features autonomously, we subsequently extended our model to include the full spectrum of available features. This expanded model featured an innovative approach for handling the high diversity of tags. Instead of one-hot encoding, we incorporated an embedding layer specifically for tags. Inspired by lecture material from the class [1], we grouped tags by movie and tokenized them to construct a vocabulary. Utilizing the pretrained GloVe weights, we initialized an embedding layer with `freeze` set to True, thereby averting the computation of gradients for these embeddings, which would otherwise slow down the training process. Each movie's tags were passed through this embedding layer to acquire the word embeddings of each tag. Our refined movie feature was then the average of these tag embeddings. This comprehensive approach, integrating a broader array of features, remarkably yielded a more effective model, reflected in an improved test RMSE of 0.800.

IV. Summary of Results

Technique	Score (RMSE)
Matrix Factorization	0.983
Matrix Factorization with SVD	0.935
Collaborative Filtering	1.390
Neural Network	0.800

TABLE I: Scores for the Movie Recommendation task

In this section we present and compare the performances of different techniques on the movie recommendation task, presented in Table I. The matrix factorization, which is our initial approach obtained an RMSE of 0.983. The matrix factorization with SVD from the surprise library achieved a score of 0.935, slightly better than our first matrix factorization.

Our second approach, collaborative filtering, gets an RMSE of 1.39, which can be considered poor compared to the other approaches.

The best score is obtained by our neural network-based approach, where we incorporated additional movie-related features to enrich the model. This advanced methodology achieved a score of 0.8. The success of the neural network-based approach reflects its effectiveness in leveraging diverse movie-related features for enhanced predictions, marking it as our top-performing strategy in this project.

V. Conclusion

In this project, we performed a comprehensive exploration of movie recommendation approaches which revealed distinct performance levels. Collaborative filtering, while a fundamental starting point, yielded a poor score compared to the other techniques. By using the Matrix factorization, we obtained more advanced predictions, and our neural network approach outperformed each of our other techniques by incorporating additional movie-related features, achieving our best score in this project.

References

[1] Aberer Karl Distributed information systems lecture material. Text classification exercises