DIS-423 Project 2 · InfoExplorers

Minh Dinh Trong, Anton Balykov, Eric Saikali

18.11.2023

---

1 Introduction

The MovieLens dataset is a widely used benchmark dataset in Recommender System, where the ratings of an user to a movie can be infered. This rating can be then used to suggest new movies to users based on their preferences. Despite being extensively researched topic, traditional recommender system methods like collaborative filtering face challenges with noisy and incomplete data. Additionally, the provided dataset lacks unique user information, making it difficult to comprehend individual preferences.

As a result, we opted for a neural network to implement the recommender system. We also utilized more data about those movies to capture dependencies with users' preferences more accurately. For this reason we used external resource and utilized the features of movies. This approach resulted in a modest MSE of 0.861. The implementation can be found at: Kaggle Link.

2 Description of Process

2.1 Roads and decision leading to our algorithm

2.1.1 Simple recommender system on initial score

Initially, as a baseline, we tried to simply allocate randomly ratings to ids. This gave us a score of 1.936. This means that any score above or near this number is issued by some poor and invalid implementation.

To extend our baseline, we tried then to compute per `movieId` the mean rating between all users. As some films are not graded yet, these films were simply given the average of all mean rating. Surprisingly this gave us a Kaggle score of 0.975 ! This clearly beats the baseline requirement and on an funny note it demonstrate that quantity of data beats the complexity of the model.

This clearly defeats the purpose of the project but for the sake of learning here is what we did :

2.1.2 Collecting and pre-processing Movie Features

We were given the genre of each movie, so we converted them to numerical data using one-hot coding: 1 if the movie follows that genre and 0 otherwise.

To extend the knowledge of each movie, we opted for scraping from online movie database. Using the efficient `tmdbv3api` package, we gathered features from The Movie Database that are highly reflect the movie's attributes: the movie overview, its popularity, and public ratings. The latter two attributes are numerical, and for the textual movie overview, we utilize a pre-trained model as used in the last project stage to obtain a numerical embedding of the content. The pre-trained model is based on Microsoft's MiniLM model [1]. This model is fine tuned on over 1 billion sentence pairs, which were obtained from Reddit Comments, Stack Exchange, Yahoo Answers, etc. [2]. From our research this model was not dependent for the task of recommender system.

The dataset also includes the `tags.csv` file, containing user comments per movie. Although this data could potentially provide insights into the user's perspective, we observed that the amount of user-movie tagging given was insufficient. Consequently, we chose to disregard this specific dataset.

2.1.3 First neural network usage

Neural networks are ideal for recommender systems due to their capacity to handle complex user-item patterns, learn from sparse data, provide personalized recommendations, and deliver state-of-the-art performance in predicting user preferences. We started with using only user's scoring of each movie.

By leveraging PyTorch's `nn.Embedding` layers, we acquire a learnable mapping of indices to embeddings for each user and movie. These embeddings are then concatenated into a 10-dimensional vector (5 for user,

5 for movie). Following this, a small MLP generates a single prediction for the movie rating. The model is optimized through minimizing the Mean Square Error between the labeled and predicted ratings.

Even though surpassing the performance of the baseline, this model design led to overfitting due to large number of parameters and not enough data to train on.

2.2 Description of Final Algorithm

To fortify the model's robustness, we added Dropout layers and incorporated weight decay parameters as means of regularization.

Additionally, we leveraged additional data about movies. We noticed that these have different ranges: public ratings range between 0 and 10 and popularity can be as high as millions. To address this discrepancy among distributions of these features, we employed a small MLP to process these additional features before concatenating them to the movie embedding.

Considering the subjective nature of user perceptions regarding a movie's plot, we implemented an attentional layer with `nn.MultiheadAttention`. This layer uses the user embedding as a query and the movie embedding as both key and value, capturing user-specific embedding for the movie plot.

Subsequently, all features are concatenated into a single vector of size 20 before being fed into the final MLP, which outputs a scalar representing the predicted ratings of the movie for the user.

Between linear layers we used activation layers 'Tanh' and 'ReLU' to inject non-linearity to the network.

The whole network is optimized using Mean Square Error between labelled and predicted rating.

2.2.1 Final Network's Process

Please find detailed neural layer sequences in the appendix.

2.2.2 Parameters

Although given a large dataset of 80668 pairs of (user, movie), the project comes with the time constraint of 10 minutes for both training and inference. Therefore, it is essential that the number of parameters should be compact. This is why input and output size are generally around 10. Besides, the dropout rate was gradually decreasing in subsequent layers, and the choice of small weight decay is to slightly encourage the model not to overfit. Finally, the selected values for input/ output channels where obtained through manual fine tuning of the model.

3 Results & Conclusion

3.1 Final Algorithm Results

As a principal evaluation, for the accuracy of the user-movie rating prediction the model obtains a score of 0.861 (Note that certain parameter choice gave 0.854) on Kaggle, with a mesured MSE of 0.639 on trained data after 30 epochs to comply with the time restrictions.

For complementary analysis, we studied score distribution :

As can be seen in the heat map/ confusion matrix (Figure 1 in the Appendix), we can see that our model tend to predict most scores accurately as the diagonal of the matrix is the boldest color.

We can also observe that the model has trouble accurately predicting extreme scores as highlighted in the distribution histograms (Figure 2 & 3). These histograms also highlight the imbalanced nature of our data set and therefore our solution. There are few low scores while the peak was between 4 and 4.5. Furthermore, the histogram of predicted rating (Figure 2) demonstrate that the model has the appearance of a Gaussian curve in term of rating distribution.

3.2 Conclusion

In this project, we developed a neural network for recommender system that implicitly handle the complexity of the data provided and derive the relationship between user-movie pairs and ratings.

We utilized movie plots, gathered from TMDB database and used embeddings to simplify interpretations of users and movies. The architecture used was simple yet robust against the noise in dataset thanks to the choice of regularization. Moreover, we used multi-head attention to let the model integrate the users' perception into the representation of the movie plot.

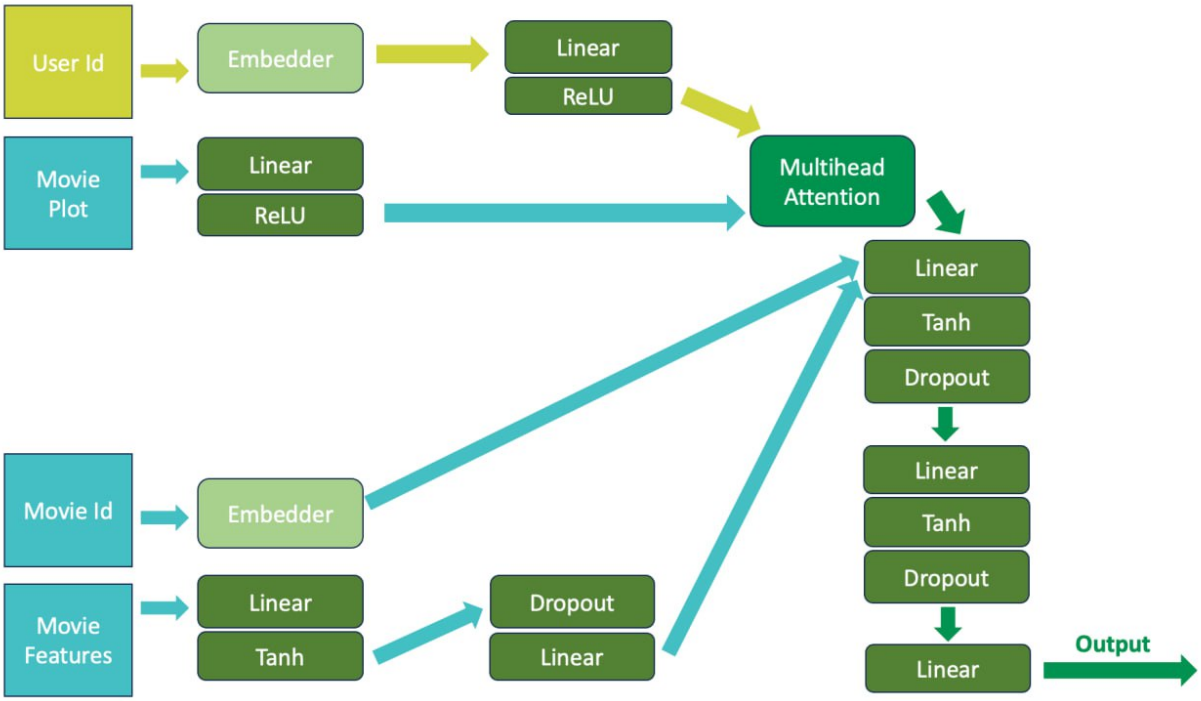
This resulted in a solid score and a robust model which is capable to capture the connections between movies and users' scores and predict new scores for given pair of user and movie.

References

[1] *Small and fast pre-trained models for language understanding and generation.* [Online]. Available: `https://github.com/microsoft/unilm/tree/master/minilm`.

[2] *Sentence-transformers/all-minilm-l6-v2 · hugging face.* [Online]. Available: `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`.

A Final neural network architecture



B Visualization of obtained prediction

B.1 Heat map for true vs. label ratings

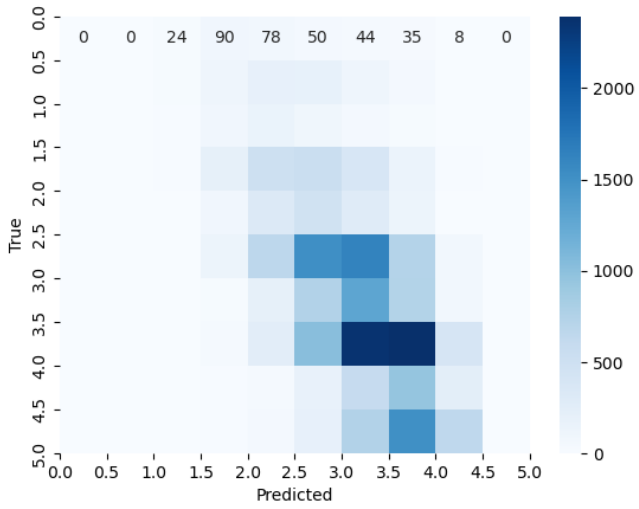


Figure 1: Heat map for true vs. label ratings

B.2 Distributions of true and label ratings

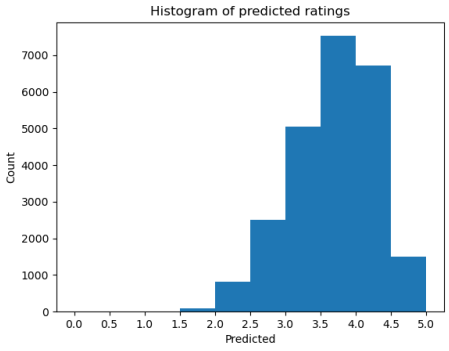


Figure 2: Histogram of predicted ratings

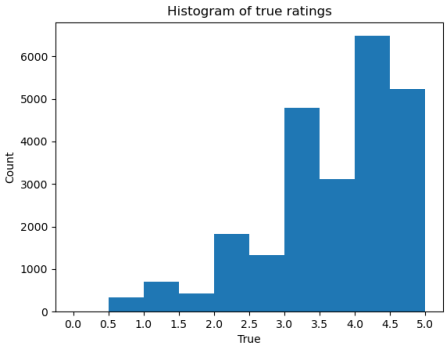


Figure 3: Histogram of true ratings