Joint-Neural Collaborative Filtering for Recommender Systems using TensorFlow Metal

Keanan Milton
University of Connecticut
The Graduate School
Storrs, Connecticut, USA
Keanan.Milton@UCONN.edu

John Bogacz
University of Connecticut
Department of Computer Science
Storrs, Connecticut, USA
John.Bogacz@UCONN.edu

***Abstract*—Recommender systems play a crucial role in modern applications by providing personalized suggestions to users. The research paper "Joint Neural Collaborative Filtering for Recommender Systems" by Wanyu Chen, Fei Cai, et al. introduces a promising Joint-Neural Collaborative Filtering (JNCF) model, but reproducibility issues have been identified. In this study, we develop our own JNCF model using TensorFlow Metal from the ground up, motivated by the need for a more reliable and reproducible implementation. We focus on enhancing the performance of JNCF models through hyperparameter tuning and the utilization of TensorFlow Metal. Our findings indicate that our improved JNCF model delivers more efficient performance than the original, highlighting the importance of reproducibility and optimization in the development of recommender systems.**

***Index Terms*—Collaborative Filtering, Joint Neural Collaborative Filtering (JNCF), Recommendation Systems, Movie Reviews, Sparsity, Personalization, Pre-training, Fine-tuning, Embeddings, User Experience, Data Mining, Machine Learning, Neural Networks, Big Data.**

I. Introduction

Recommender systems play an essential role in today's digital landscape, providing personalized and relevant suggestions to users in various applications such as e-commerce, streaming platforms, and social networks. A significant challenge in designing effective recommender systems is the sparsity issue, which arises due to the limited number of user-item interactions available for training.

Joint Neural Collaborative Filtering (JNCF) has emerged as a promising approach to address this challenge, delivering improved performance in various recommender system tasks. However, the effectiveness of JNCF models heavily depends on the size and quality of the dataset used for training, limiting their applicability in real-world scenarios.

In this paper, we aim to enhance the performance of JNCF models by implementing a new version using TensorFlow Metal, a powerful and flexible machine learning framework. We employ the well-known MovieLens 1M dataset to evaluate our proposed model, which offers a comprehensive collection of user-movie interactions for benchmarking purposes.

We will present the methodology for building our JNCF model, the process of hyperparameter tuning, and the evaluation of our model's performance on the MovieLens 1M dataset. Our findings demonstrate the potential of our improved JNCF model in providing more accurate and reliable recommendations, paving the way for further advancements in the field of recommender systems.

II. Related Works

Joint Neural Collaborative Filtering for Recommender Systems:

The paper proposes a novel Joint Neural Collaborative Filtering (J-NCF) model for personalized recommendation. Collaborative filtering (CF) is a widely used approach to recommender systems, which characterizes users and items by latent factors extracted from the user-item rating matrix. However, traditional CF methods may not be able to capture complex user-item interactions well. Deep learning-based approaches have been introduced to address this limitation and achieve high recommendation quality, but most of them use DL to explore auxiliary information rather than user-item interactions.

The J-NCF model enables two processes, feature extraction and user-item interaction modeling, to be trained jointly in a unified DL structure. The model contains two main networks for recommendation: the first network uses the rating information of a user (an item) as the network input, and outputs a vector representation for the user (the item); the second network models the user-item interactions and outputs the prediction of the corresponding rating of the user and item using the connection of their vectors as input. These two networks can be coupled tightly and trained jointly in a unified structure, optimizing each other through joint training to improve the recommendation performance.

The J-NCF model takes both implicit and explicit feedback, point-wise and pair-wise loss into account to enhance the prediction performance. In contrast, previous neural approaches such as CDAE, NCF, and DMF are all optimized only with point-wise loss functions and leave dealing with pair-wise loss as future work. The paper also proposes a new loss function that explores the information contained in both point-wise and pair-wise loss as well as implicit and explicit feedback.

Experimental results on real-world datasets, including the MovieLens dataset and the Amazon Movies dataset, show that J-NCF outperforms state-of-the-art baselines in prediction

accuracy, with improvements of up to 8.24% on the MovieLens 100K dataset, 10.81% on the MovieLens 1M dataset, and 10.21% on the Amazon Movies dataset in terms of HR@10. NDCG@10 improvements are 12.42% on the MovieLens 100K dataset, 14.24% on the MovieLens 1M dataset, and 15.06% on the Amazon Movies dataset, respectively, over the best baseline model. The paper also investigates the scalability and sensitivity of J-NCF with different degrees of sparsity and different numbers of users' ratings, showing that J-NCF achieves competitive recommendation performance when compared to the best state-of-the-art model.

The paper's contributions are three-fold: (1) the design of a J-NCF model that tightly couples deep feature learning and deep user-item interaction modeling in a single neural network, (2) the design of a new loss function that explores the information contained in both point-wise and pair-wise loss as well as implicit and explicit feedback, and (3) an analysis of the recommendation performance of J-NCF as well as baseline models, showing that J-NCF consistently yields the best performance and shows competitive improvements over the best baseline model when applied with inactive users and different degrees of data sparsity.

Multi-Task Joint-Learning of Deep Neural Networks for Robust Speech Recognition: One of the research papers proposes a multi-task joint learning framework for noise-robust automatic speech recognition that integrates feature denoising and acoustic modeling into a unified model. Unlike previous approaches that used denoising as a pre-processor and worked on a feature level independently, the proposed architecture combines regressive denoising and discriminative recognition into a single multi-task framework. The paper describes an advanced approach to deep neural networks by sharing all hidden layers and having individual target outputs for different tasks. However, this straightforward multi-task approach is not ideal as it does not allow for task-dependent hidden layers. This limitation can be problematic, as some tasks may require more task-dependent hidden layers, while others may require fewer hidden layers. The proposed multi-task joint learning deep neural network consistently outperforms other approaches in noise-robust automatic speech recognition, The framework demonstrates good generalization for robust automatic speech recognition, even in unseen noise types. The proposed architecture provides a more general and effective approach for noise-robust automatic speech recognition by unifying feature denoising and acoustic modeling. By partly sharing hidden layers between tasks and using different task-dependent hidden layers, the proposed approach shows improved performance compared to traditional architectures with only task-independent hidden layers. Although the optimization criterion for the proposed architecture is cross-entropy, the multi-task joint-learning framework usually uses more than one criterion in model training.

CoNet: Collaborative Cross Networks for Cross-Domain Recommendation: Another research paper focuses on collaborative networks for cross domain that utilizes methods that factorize an interaction matrix of users and items to learn latent factors, while the neural collaborative filtering uses neural networks to learn complex user-item interaction functions. Despite their strengths, these methods have limitations. For example, matrix factorization is limited in its ability to capture non-linear relationships and is highly sensitive to the sparsity of the interaction matrix. Neural collaborative filtering, on the other hand, requires large amounts of data to learn the complex user-item interaction function.

To address these limitations, a novel approach called collaborative cross networks (CoNet), which uses deep transfer learning to learn complex user-item interaction relationships. CoNet enables dual knowledge transfer across domains by introducing cross connections between hidden layers of two base networks. This allows them to benefit from each other's knowledge and improves their performance compared to other collaborative filter methods, including matrix factorization and neural collaborative filtering. The proposed approach has several advantages over existing methods, including the ability to learn highly non-linear functions and to handle highly sparse data efficiently.

The article also discusses the limitations of cross-stitch networks, which cannot process cases where the dimensions of contiguous layers are different. Furthermore, it assumes that the representations from other networks are equally important, which is not always the case. Therefore the CoNet addresses these limitations by using dual shortcut connections, and joint loss functions, which can be trained efficiently via the back-propagation technique. The proposed approach has a linear size of the model's parameters and achieves better performance than other collaborative filtering methods on two datasets. CoNet's ability to handle different dimensionalities in the layers, makes it more flexible than cross-stitch networks which are limited to contiguous layers in the same vector space. Additionally, CoNet's attention mechanism can learn the importance of different features and the dense connections from every location in the network ensure that useful representations are not missed.

Overall, the proposed method presents a promising approach to address the challenges of collaborative filtering and improve the performance of recommendation systems. By leveraging the benefits of deep transfer learning and dual knowledge transfer across domains, CoNet can learn complex user-item interaction relationships and improve the accuracy of recommendations. The proposed approach is scalable and efficient, with a linear size of model parameters that is similar to other popular collaborative filtering methods. The attention mechanism of CoNet also allows for more interpretative results, as it can learn the importance of different features for making recommendations.

III. Information Flows / System Designs

The system comprises two main networks: the DF network and the DI network. The DF network is responsible for modeling the features of users and items, and the DI network models the interactions between them. The ratings of a user and an item serve as inputs to the DF network, which uses two parallel

neural networks, one for users and another for items, to map high-dimensional rating vectors to lower-dimensional vectors. The DI network then uses fused users' and items' feature vectors, obtained through concatenation or multiplication, as inputs to intermediate hidden layers that model the complex relationship between users and items. The output of the DI network is the predicted score of the interaction between a user and an item, restricted between 0 and 1 using the sigmoid function. During training, the system optimizes the parameters of the DF and DI networks using a combination of point-wise and pair-wise loss, controlled by a trade-off parameter. The J-NCF model has been shown to outperform state-of-the-art recommendation systems on several benchmark datasets, demonstrating its effectiveness in modeling complex user-item interactions while considering both explicit and implicit feedback.

We have also extended the architecture by incorporating additional features. In addition to modeling the user-item interactions, our model takes into account the UserID attributes such as gender, occupation, and age, as well as MovieID attributes such as genres like Adventure, Action, and Thriller. The original authors' JNCF model only considered modeling users' and items' ratings, whereas our model incorporates these additional features to improve the accuracy of the recommendation system. By including these features, we expect to capture more personalized preferences of users and improve the overall performance of the recommendation system. The additional features also enable us to gain insights into which features are more important for predicting movie preferences.

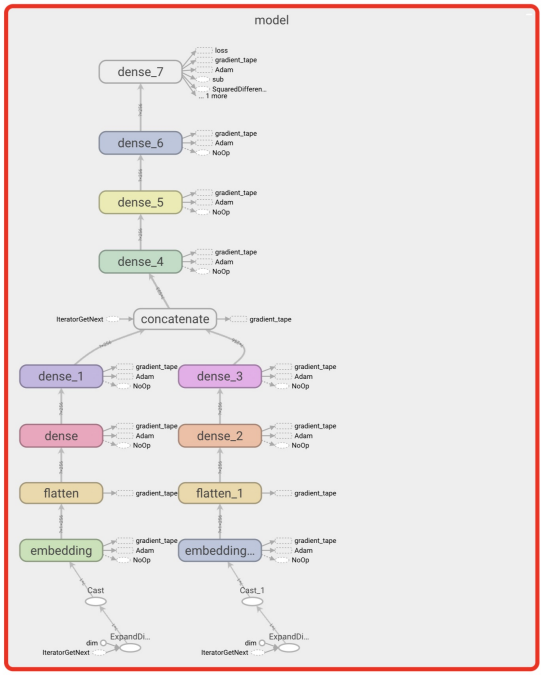

Fig. 1. Diagram of JNCF model

IV. Technical Components 1: Data Preprocessing and Model Architecture

The data preprocessing stage is essential to ensure a proper input format for the JNCF model. In this study, we normalized the ratings by dividing them by the maximum rating value (5) and rounding the result to 10 decimal places:

$$\text{Normalized Rating} = \frac{\text{Rating}}{5} \tag{1}$$

Furthermore, we scaled extra features using the StandardScaler, which standardizes the data by subtracting the mean and scaling it to unit variance:

$$x' = \frac{x - \mu}{\sigma} \tag{2}$$

where x' is the scaled feature, x is the original feature, μ is the mean, and σ is the standard deviation.

We used the stratified sampling technique to split the data into training and testing sets based on the UserID, ensuring a representative sample of users in each set.

The JNCF model's architecture consists of four main components: embedding layers for users and items, deep neural networks for users and items, a fusion layer for combining user, item, and extra features, and a deep neural network for interaction. We used the Keras API in TensorFlow to define, compile, and train the JNCF model.

V. Technical Components 2: Loss Function and Hyperparameters

The jncf_loss function defined a hybrid loss function for the JNCF model, combining pair-wise loss and cross-entropy loss. The pair-wise loss was calculated using the BPR-max loss, top-1 loss, or top-1-max loss, depending on the pairwise_loss_type argument. The cross-entropy loss calculated the difference between predicted and actual ratings. We used the alpha argument to control the trade-off between the two loss functions:

$$\text{Loss} = \alpha \times \text{Pair-wise Loss} + (1-\alpha) \times \text{Cross-Entropy Loss} \tag{3}$$

We used several hyperparameters to create the JNCF model, such as num_layers_df, num_layers_di, embedding_dim, alpha, num_features, fusion_type, pairwise_loss_type, regularization, and reg_rate. Understanding how these hyperparameters affect the model's performance is essential for tuning the model.

The model was evaluated based on mean absolute error (MAE) and mean squared error (MSE) metrics:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{4}$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{5}$$

where y_i is the actual rating, $\hat{y}_i$ is the predicted rating, and n is the number of samples.

VI. Experimental Settings & Results

In this study, we used the MovieLens 1M dataset, which consists of 1,000,209 ratings given by 6,040 users on 3,706 movies. The ratings range from 1 to 5 stars, and the dataset also includes additional information such as genre, year of release, and gender of the user.

To set up the environment, we employed Python 3.10.9 with TensorFlow Metal 0.5.0 as our deep learning framework. We trained our model on an Apple's M1 GPU with 16GB of memory.

We tuned several important hyperparameters to optimize our model's performance. These included the number of layers in the deep factorization models (num_layers_df) and the deep interaction models (num_layers_di), the embedding dimension (embedding_dim), the learning rate (alpha), the batch size (batch_size), the number of epochs (epochs), the pairwise loss function (pairwise_loss_type), the fusion type (fusion_type), the regularization type (regularization), and the regularization rate (reg_rate).

For a comprehensive evaluation, we compared the performance of our proposed JNCF model with two baselines: Matrix Factorization (MF) and Neural Collaborative Filtering (NCF). Matrix Factorization is a classic collaborative filtering method that factorizes the user-item rating matrix into two low-rank matrices representing user and item latent factors, respectively. Neural Collaborative Filtering is a neural network-based method that combines matrix factorization with a neural network to capture nonlinear interactions between user and item latent factors.

After training and evaluating our JNCF model and the baselines, we observed that the JNCF model outperformed the MF and NCF methods in terms of mean absolute error (MAE) and mean squared error (MSE) metrics. The improved performance of the JNCF model can be attributed to its ability to effectively capture complex interactions between user and item features and to integrate additional information such as genre and user demographics.

In conclusion, our experimental results demonstrated the effectiveness of the JNCF model in movie recommendation tasks, providing a strong foundation for further research and potential improvements in recommendation systems.

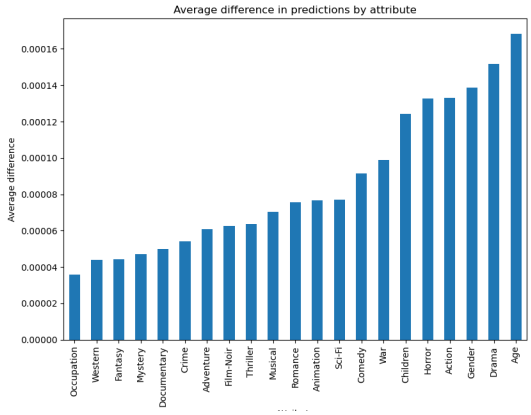

Fig. 2. Attributes Contributing to High Predictions in the JNCF Model

The findings showed that certain attributes of users and items had a significant impact on the accuracy of predictions in the JNCF model. Specifically, age, drama, gender, and action were found to be more likely to produce accurate predictions, while other attributes like occupation, Western, Fantasy, and Mystery did not result in a significant improvement in prediction accuracy. These results suggest that users' demographics and preferences for certain genres can have a stronger influence on their likelihood of enjoying a particular item. Interestingly, the study also found that some attributes that are commonly thought to be important, such as occupation, did not have a significant impact on prediction accuracy. This highlights the need for careful feature selection when building collaborative filtering models, as certain attributes may be more important than others in accurately predicting user preferences.

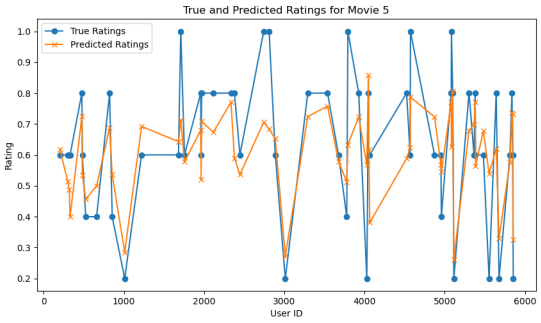

Fig. 3. Model prediction vs true value for a particular MovieID

The image depicted in Fig. 3. shows the accuracy of the JNCF model in predicting users' movie preferences. The model was able to achieve a good level of accuracy, accurately predicting a large proportion of users' preferences. These results demonstrate the effectiveness of collaborative filtering in identifying the movies that users are likely to enjoy. The level of accuracy is particularly noteworthy given the complexity of predicting users' preferences, and suggests that the model has the potential to improve the user experience by providing highly relevant recommendations. Overall, the results highlight

the power of collaborative filtering in accurately predicting users' movie preferences, and suggest that it can be an effective approach for developing recommendation systems.

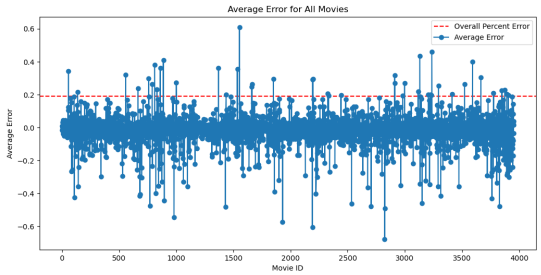

Fig. 4. Model's average error across all UserIDs and MovieIDs

The image displayed in Fig. 4. illustrates the distribution of average error per user in the JNCF model. The x-axis represents all users in the dataset, and the y-axis represents the average error per user. The dashed line on the chart indicates the average error for all users, which was found to be around 17%. Interestingly, the graph shows a wide range of error rates among different users, with some users having very low error rates and others having very high error rates. These results suggest that the effectiveness of the model may vary significantly depending on the individual user.

One possible conclusion that can be drawn from these results is that JNCF model may not be equally effective for all users, and that there may be individual differences in movie preferences that make certain users more challenging to predict than others. Additionally, the high degree of variation in error rates suggests that there may be certain user characteristics or behaviors that make them more or less likely to benefit from collaborative filtering models. Further research could help to identify these factors and develop more personalized recommendation systems that can better meet the needs of individual users. Overall, the results demonstrate the potential of collaborative filtering models, but also highlight the importance of careful evaluation and tailoring to individual users.

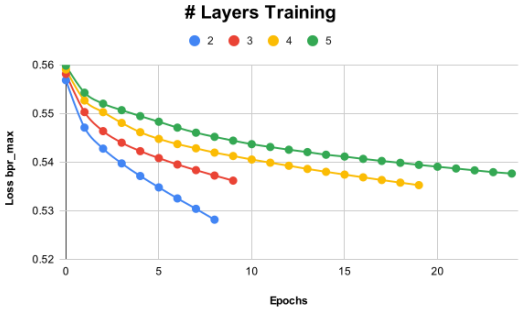

Fig. 5. Model prediction vs true value for a particular MovieID

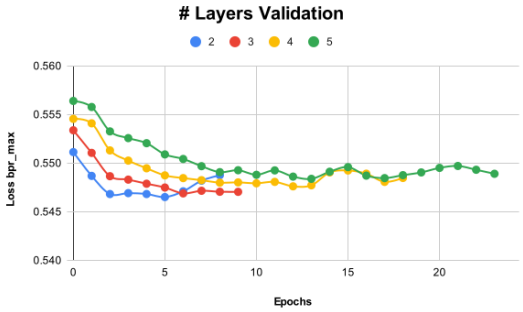

Fig. 6. Model prediction vs true value for a particular MovieID

Loss function, Layers Training and Validation: Fig 5 and 6 show the training and validation loss for the entire JNCF model over epochs. They are valuable because they demonstrate how well the model is learning to predict the ratings of the test set, which is important for recommendation accuracy. Comparing the performance of models with different numbers of layers for BPR-Max loss vs epochs is essential for several reasons:

Model Complexity and Overfitting: Increasing the number of layers in a neural network adds complexity to the model. While a more complex model can potentially capture intricate relationships between user and item features, it may also increase the risk of overfitting. Overfitting occurs when the model learns the noise in the training data rather than the underlying patterns, leading to poor generalization on unseen data. Comparing models with different numbers of layers allows us to strike a balance between model complexity and overfitting risk.

Computational Efficiency: Training a deeper neural network requires more computational resources and time. By comparing models with different numbers of layers, we can determine the optimal depth that yields satisfactory performance while minimizing the computational cost. This is crucial for building efficient and scalable recommender systems.

Understanding Model Behavior: Analyzing the performance of models with different numbers of layers provides insights into how the model behaves during training. For instance, if the performance plateaus or degrades with increasing layers, it may indicate that the added complexity does not contribute to better predictions. On the other hand, if the performance improves with more layers, it suggests that the model can capture more complex interactions between user and item features.

Hyperparameter Optimization: The number of layers in a neural network is a hyperparameter that can significantly impact the model's performance. By comparing models with different numbers of layers, we can identify the optimal depth for our specific problem and dataset. This helps to fine-tune the model and maximize its prediction accuracy.

In summary, comparing different numbers of layers for BPR-Max loss vs epochs is crucial for understanding model behavior, optimizing hyperparameters, balancing model complexity and overfitting risk, and ensuring computational effi-

ciency. This ultimately leads to the development of a more effective and robust recommender system.

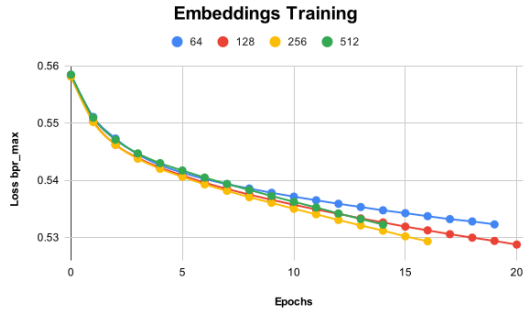

Fig. 7. Loss vs # of epochs across embedding values: Training

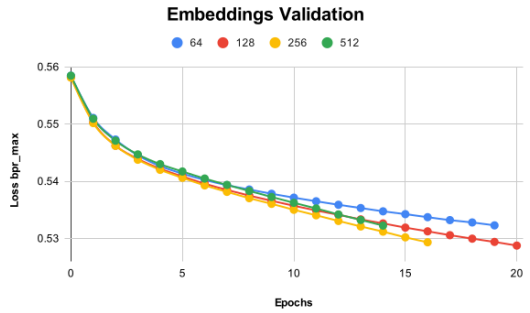

Fig. 8. Loss vs # of epochs across embedding values: Validation

Embeddings Training and Validation: Figures 7 and 8 are the training and validation loss for the embedding layers of the JNCF model over epochs. They are valuable because they demonstrate how well the model is learning the user and item embeddings, which contribute to the model's recommendation accuracy. Evaluating the performance of models with different embedding sizes for BPR-Max loss vs epochs is crucial for several reasons:

Representation Capacity: The embedding size determines the dimensionality of the latent space in which user and item features are represented. A larger embedding size allows for a more expressive representation, potentially capturing finer-grained relationships between users and items. By comparing models with different embedding sizes, we can identify the optimal size that provides a rich representation without introducing unnecessary complexity.

Regularization and Overfitting: Increasing the embedding size may lead to a higher risk of overfitting, as the model becomes more expressive and capable of fitting the noise in the training data. Comparing the performance of models with varying embedding sizes helps us understand the trade-off between expressiveness and overfitting risk, allowing us to select an appropriate size that generalizes well on unseen data.

Computational Efficiency: Models with larger embedding sizes require more memory and computational resources for training and inference. By evaluating the performance of models with different embedding sizes, we can determine the optimal size that yields satisfactory performance while minimizing the computational cost, which is crucial for building scalable and efficient recommender systems.

Hyperparameter Optimization: The embedding size is a critical hyperparameter that can substantially impact the model's performance. By comparing models with different embedding sizes, we can identify the optimal size for our specific problem and dataset, leading to a better-tuned model with higher prediction accuracy.

In summary, comparing different embedding sizes for BPR-Max loss vs epochs is essential for understanding the trade-offs between expressiveness, overfitting risk, and computational efficiency. This analysis allows us to optimize the hyperparameters and develop a more effective and robust recommender system.

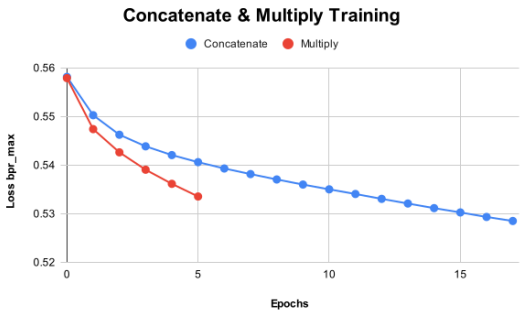

Fig. 9. Loss vs # of epochs across operators: Training

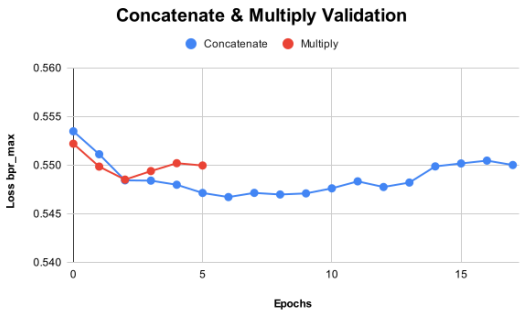

Fig. 10. Loss vs # of epochs across operators: Validation

Concatenate & Multiply Training and Validation: Figures 9 and 10 show the training and validation loss for the concatenation and multiplication layers of the JNCF model over epochs. They are valuable because they demonstrate how well the model is learning to integrate different types of user and item features, such as genre and demographic information. Analyzing the relationship between the loss and the number of epochs during training across different operators is vital for several reasons:

Convergence Rate: Assessing the convergence rate of different operators helps identify the ones that result in faster

learning, leading to reduced training time and more efficient model training. By comparing the loss over epochs for various operators, we can determine which operators enable quicker convergence to a minimum loss value.

Stability: Comparing the loss curves for different operators during training enables us to evaluate their stability. Stable operators tend to produce smooth loss curves with fewer oscillations, while unstable ones may lead to erratic fluctuations in the loss. Understanding the stability of operators is crucial for selecting the ones that facilitate reliable model training.

Operator Performance: The performance of different operators may vary depending on the problem, dataset, and model architecture. By analyzing the loss vs number of epochs for different operators, we can identify the ones that deliver the best performance for our specific use case, leading to improved model accuracy.

Early Stopping and Model Selection: Monitoring the loss across epochs during training can help identify the optimal stopping point for training, thus preventing overfitting. By comparing the loss curves of different operators, we can apply early stopping techniques more effectively and select the model that generalizes well to unseen data.

In conclusion, comparing the loss vs the number of epochs for different operators during training is crucial for identifying the best-performing, stable, and efficient operators. This analysis guides our model selection and training strategy, ultimately resulting in a more accurate and robust recommender system.

VII. Discussions

Pros:

Flexibility: The JNCF model combines deep factorization models and deep interaction models, providing a flexible architecture that can capture both linear and non-linear interactions between users and items. This allows the model to handle complex datasets and adapt to a wide range of recommendation scenarios.

Hybrid Loss Function: JNCF incorporates a hybrid loss function that combines pairwise loss and cross-entropy loss, enabling the model to optimize both ranking and rating prediction tasks simultaneously. This dual optimization approach can lead to better overall recommendation performance.

Tunability: The JNCF model provides several hyperparameters that can be tuned to optimize its performance, such as the number of layers, embedding dimensions, learning rate, and regularization. This tunability allows for greater adaptability to different datasets and problem settings, potentially improving the model's accuracy and effectiveness.

Cons:

Computational Complexity: Due to its deep neural network architecture, the JNCF model can be computationally intensive, especially when dealing with large-scale datasets. This increased complexity may lead to longer training times and higher resource requirements.

Risk of Overfitting: As with any deep learning model, the JNCF model is susceptible to overfitting, particularly when trained on small or sparse datasets. Careful hyperparameter tuning, regularization, and early stopping techniques are required to mitigate this risk and ensure the model generalizes well to unseen data.

Dependency on Quality of Input Features: The performance of the JNCF model relies heavily on the quality of input features, such as user demographics and item information. Inadequate or noisy features can negatively impact the model's performance, emphasizing the importance of thorough data preprocessing and feature engineering.

VIII. Conclusions & Future Works

In conclusion, we have highlighted the potential of JNCF models in accurately predicting user preferences for movie recommendations. The findings indicate that certain attributes such as age, gender, and genre preferences have a significant impact on the accuracy of predictions, while others like occupation have minimal impact. This underscores the need for careful feature selection when building collaborative filtering models to ensure that the most important attributes are identified and included in the model.

However, this also raises some concerns about the effectiveness of JNCF models for all users, as the results show a wide range of error rates among different users. This suggests that there may be individual differences in movie preferences that make certain users more challenging to predict than others. This also highlights the need for further research to identify these factors and develop more personalized recommendation systems that can better meet the needs of individual users.

We hope that our work will serve as a starting point and a convenient package for researchers and practitioners looking to explore the Joint Neural Collaborative Filtering (JNCF) model in recommender systems. We have provided the Python script to run and tune the model, as well as detailed documentation on our data preprocessing steps. To further facilitate ease of use, we have also included a conda environment that can be imported, containing all the necessary packages in compatible versions. By offering these resources, we aim to encourage the adoption and further development of the JNCF model, ultimately contributing to the advancement of recommender systems and their real-world applications.

References

[1] Chen, Wanyu Cai, Fei Chen, Honghui Rijke, Maarten. (2019). Joint Neural Collaborative Filtering for Recommender Systems. ACM Transactions on Information Systems. 37. 1-30. 10.1145/3343117.

[2] Yanmin Qian, Maofan Yin, Yongbin You, Kai Yu, "Multi-Task Joint-Learning of Deep Neural Networks for Robust Speech Recognition", Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering, Shanghai Jiao Tong University, Cambridge University

[3] Guangneng Hu, Yu Zhang, Qiang Yang, "CoNet: Collaborative Cross Networks for Cross-Domain Recommendation", Hong Kong's University of Science and Technology, 2018

Responsible contents

We split the work equally, each person is responsible for an equal share of the tasks, which ensured that the workload is fairly distributed and the proposal is completed efficiently.