Received November 1, 2021, accepted November 20, 2021, date of publication November 23, 2021,
date of current version December 6, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3130369

DNR: A Unified Framework of List Ranking With Neural Networks for Recommendation

CHUNTING WEI, JIWEI QIN, AND WEI ZENG
School of Information Science and Engineering, Xinjiang University, Urumqi 830046, China
Key Laboratory of Signal Detection and Processing, Xinjiang Uygur Autonomous Region, Xinjiang University, Urumqi 830046, China

Corresponding author: Jiwei Qin (jwqin_xju@163.com)

This work was supported by the Science Fund for Outstanding Youth of Xinjiang Uygur Autonomous Region under Grant No. 2021D01E14, the National Science Foundation of China under Grant No. 61867006, The Major Science and Technology Project of Xinjiang Uygur Autonomous Region under Grant No. 2020A03001, the Innovation Project of Sichuan Regional under Grant No. 2020YFQ2018 and the Key Laboratory Open Project of Science & Technology Department of Xinjiang Uygur Autonomous Region named Research on Video Information Intelligent Processing Technology for Xinjiang Regional Security.

ABSTRACT In recent years, list ranking learning has attracted further attention in the field of recommendation. Most of the existing list ranking methods use traditional or deep learning methods to fit user-item interactions and output a personalized ranking list. These solutions mainly focus on a certain aspect of linear and nonlinear interactions when learning latent structures of users and items. In fact, linear interactions and nonlinear interactions coexist and play complementary roles in a recommendation system. Once any type of interaction is ignored, it will result in the loss of linear or nonlinear features, which will further affect the overall performance of the model. To address this problem, we propose a general framework, DNR, short for Deep Neural Rank. To jointly learn linear and nonlinear features, it uses both traditional and deep learning methods to fit user-item interactions. This is a flexible architecture that can not only be extended to the integration of various linear models and nonlinear models but also be simplified for pairwise learning to rank. This paper focuses on the feasibility of integrating matrix factorization (MF) and multi-layer perceptron (MLP) into the framework as linear and nonlinear methods. Eventually, we conduct extensive experiments on three data sets (Movielens100K, Movielens1M, and Yahoo Movies). The results show that our proposed DNR framework achieves a significant improvement compared to existing methods.

INDEX TERMS Recommender systems, collaborative filtering, deep learning, neural networks.

I. INTRODUCTION

With the rapid development of information technology, users are facing a huge number of information choices [1]–[3]. For general users, it is difficult to obtain the required information in a short period of time. To alleviate information overload and meet the diverse needs of users, personalized recommendation systems have been developed and are beginning to play important roles in modern society [4], [5].

Recently, deep learning models have been widely used in recommendation systems to improve the quality of recommendations [1], [25]–[27]. For example, Zhang *et al.* [6] jointly learned explicit and implicit couplings between users and item features and adopted neural coupling learning for collaborative filtering (CoupledCF). Jiang *et al.* [8] presented a learning model of knowledge representation based on the self-attention mechanism, which uses the graph structure of knowledge graph and self-attention mechanism to represent entity learning features and handle complex relationship feature representation learning. These examples indicate the effectiveness of using deep learning models for collaborative recommendation.

However, the single use of deep learning models may lose a large number of linear features in user-item interactions and make the model learning process inefficient. For traditional linear models, they mostly use dot products to estimate matching scores. Although this low-rank relationship can retain the user-item linear features, it artificially limits the model learning similarity. These linear methods ignore the high-rank relationship and user-item nonlinear features.

The associate editor coordinating the review of this manuscript and approving it for publication was Paolo Crippa.

According to the above discussion, either of these two methods will cause the loss of entity features and limit the overall performance of the model.

To solve this problem, we propose DNR, a list ranking method based on deep neural networks. DNR combines traditional and deep learning methods and considers the strengths of linear and nonlinear models to perform stronger and more robust fitting of user-item interactions. By using a point-by-point approach, DNR focuses on predicting the accurate score of each item. In practice, users mostly care more about the rankings of items than their accurate ratings. For instance, when users select a movie in a recommended playlist, they often do not care about the score of each movie but try the first one. Therefore, the final goal of DNR is to rank each item in the list. In addition, since the amount of implicit data in the real world is large and can more accurately reflect user preferences [12], we focus on implicit feedback in this article.

Then, we simplified the model from the top-n listwise to a simpler structure: the pairwise method. In this method, each user is expressed as a set of preferences for items, which is more helpful for users to understand their preferences [9], [10]. In the experimental part of our paper, we also compare the different characteristics of the pairwise method and the listwise method through experiments and discuss how to choose between the two in practice.

In summary, our main contributions are as follows:

1. We present a new type of neural network architecture to fit the linear and nonlinear interaction process of users and items and devise the DNR framework for list learning based on neural networks.

2. We show that the top-n listwise ranking learning can be simplified to pairwise ranking learning for efficiency and compare the different characteristics of the two.

3. We explore the impact of the fusion of linear and nonlinear models on hidden layers through extensive experiments.

4. We conduct extensive experiments on three real-world data sets to prove the effectiveness of our DNR method and discuss future improvements.

II. RELATED WORKS

A. TRADITIONAL RANKING METHODS

Due to the distinguishing capability of utilizing collective wisdom and experience, traditional matrix factorization (MF) methods have been widely used to build recommender systems [13]–[15]. Their core idea is to learn the latent user-item interactions in a low dimension from the interaction matrix. Zheng *et al.* [16] proposed a gradient boosting rank (GBrank) framework, which uses a pairwise method to construct a special loss function for training. Rendle *et al.* [17] proposed a Bayesian ranking (BPR) framework that uses implicit feedback from users (such as clicks, favorites, shopping carts, etc.) to rank items through the maximum posterior probability obtained by Bayesian analysis. Park *et al.* [18] use auxiliary information, such as user statistics and item content characteristics, to solve the cold start problem. Shi *et al.* [19] proposed a combination of a list-based learning ranking algorithm and matrix factorization, ListRank-MF, which optimizes the model for recommendation by minimizing a loss function.

Although these methods perform well, most of them overlook the key problem. They only focus on the linear interaction between the user and the item and cannot capture higher-dimensional information and more complex structures from user-item interactions.

B. RANKING METHODS BASED ON DEEP LEARNING

In recent years, due to the powerful ability of deep learning models to learn complex latent features, many researchers have developed recommendation systems using deep learning models [12]. For example, Cheng *et al.* [20] presented the Wide & Deep framework, which learned the matching function from input continuous features and categorical features of users and items through logistic regression (LR) and MLP. He *et al.* [21], by replacing the inner product with a neural architecture that can learn an arbitrary function from data, presented a general framework named neural collaborative filtering (NeuMF). Guo *et al.* [12] proposed the deep factorization machine (DeepFM) model, which combines the feature learning capabilities of factorization machines and deep learning in a new neural network structure. Chen *et al.* [29] proposed the co-occurrence neural networks (CoNet) model, which models the co-occurrence of items and uses an attention network to learn user preference weights for different items.

Although quite a few deep learning models have been widely used for recommendation, there are relatively few studies on ranking learning among them. In fact, these ranking methods have a wide influence and inspired our method. For example, Burges *et al.* [22] presented a ranking method, RankNet, which is a pairwise method with a neural network that can calculate the probability of a user's pairwise preference. Chen *et al.* [11] proposed the deep neural rank (DeepRank) model, which uses the MLP to fit user-item interactions and proves the effectiveness of pairwise ranking. Deng *et al.* [7] presented a list ranking framework, deep collaborative filtering (DeepCF), which uses a deep neural network for implicit feedback to overcome the limited expressiveness of dot product and the weakness in capturing low-rank relations. Cai *et al.* [30] proposed category-aware collaborative sequential recommender (CoCoRec), which leverages category information to capture the context-aware action dependence for ranking recommendation.

Deep learning models can capture the complex and in-depth information of the interaction between users and items and enrich the deep representation of users and items. However, the single use of deep learning models to fit user-item interactions limits the learning of linear interactions. This is a considerable loss to the expressive ability of the entire model.

III. OUR APPROACH

We first show the general process of the DNR framework and introduce the selected linear layer and nonlinear layer. Then, we demonstrate how to integrate these two layers in our DNR framework and how to learn the final model. Finally, we introduce a method to simplify DNR into a pairwise ranking model.

A. GENERAL PROCESS

Figure 1 shows the general process of the DNR. The linear layer and the nonlinear layer learn independent embeddings from the input layer and then use the linear model and the nonlinear model to process embeddings, respectively. To obtain the matching score, we fuse these two layers and import them into a fully connected layer, enabling the model to assign different weights to features. Finally, by mapping the score into a probability, we can rank each item in the list.

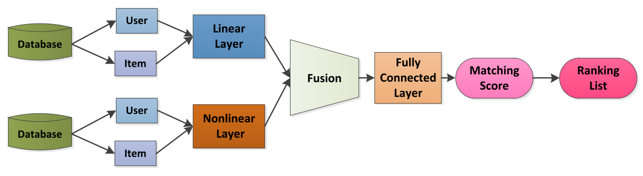

FIGURE 1. The general process of the DNR framework.

B. LINEAR LAYER

A linear layer is used to learn the linear interaction between users and items with linear models. It has high flexibility and can be extended to the MF model, singular value decomposition (SVD) model, or factorized metric learning (FML) model. The MF model is currently the most widely used linear model in recommendation systems [23]. To save space, this paper focuses on the feasibility of integrating the MF model into DNR.

The MF method focuses on decomposing the rating matrix into the product of two low-rank matrices. Through the decomposition, the user and the item are embedded into the same k-dimensional vector space, and the inner product of the user vector and the item vector represents the user's preference for the item. Considering that this article focuses on implicit feedback, we can set the value of the true rating x_{ui} as a label − 1 means there is the interaction between user u and item i, and 0 otherwise. Let p_u and q_i denote the latent vector of user u and item i; and $\hat{x}_{ui}$ denotes the predicted rating of item i given by user u:

$$\hat{x}_{ui} = \mathrm{p}_u^T \mathrm{q}_i = \sum_{k=1}^{K} p_{uk} q_{ik}, \tag{1}$$

where K is the dimension of the latent vector space. Here, we assume that each dimension of the latent space is independent and combine them linearly into the same weight, so MF can be considered a linear model of latent factors.

Matrix factorization is one of the most popular methods in recommendation systems [10], but the inner product of matrix factorization does not satisfy the triangle inequality, which will reduce the fine-grained similarity between users and items and limit the expressiveness of matrix factorization. This leads to suboptimal solutions. More importantly, the single use of matrix factorization will ignore many nonlinear features and cause certain errors. This conclusion will be proven in the experimental part.

C. NONLINEAR LAYER

Here we choose a multi-layer perceptron (MLP) as the nonlinear layer. In the DNR framework, we add hidden layers to the entity embedding layer and use a standard MLP to learn nonlinear interactions between the latent features of users and items. The standard MLP model architecture is shown in Figure 2.

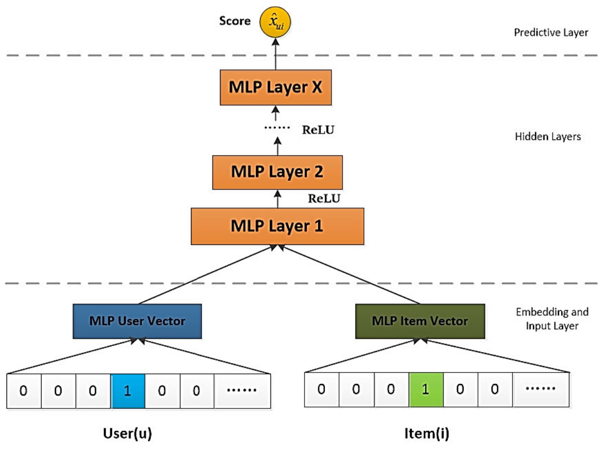

FIGURE 2. The standard architecture for MLP.

1) EMBEDDING AND INPUT LAYERS

The function of the MLP embedding layer is to transform users and items into dense vectors in a low-dimensional space, that is, user vectors and item vectors. In addition, embedding can express more implicit semantic relations, further improving the generalization ability.

2) HIDDEN LAYERS

The hidden layer is a stack of several fully connected layers. Its function is to perform joint coding of user embeddings and item embeddings to capture the high-dimensional characteristics of users and items and to fit the nonlinear interactions and certain underlying structures between them. We only use two hidden layers in DNR, which simplifies the model, reduces the difficulty of parameter adjustment, and achieves better generalization. This point of view will be proven in later experiments. Each layer of MLP under the DNR framework can be defined as:

$$\begin{aligned} x_1 &= k_1(\mathrm{p}_u, \mathrm{q}_i) = \begin{bmatrix} \mathrm{p}_u \\ \mathrm{q}_i \end{bmatrix}, \\ x_2 &= k_2(x_1) = a_2\left(\mathrm{W}_2^T x_1 + \mathrm{b}_2\right), \\ &\cdots\cdots \\ x_L &= k_L(x_{L-1}) = a_L\left(\mathrm{W}_L^T x_{L-1} + \mathrm{b}_L\right), \end{aligned} \tag{2}$$

where a_l, b_l, and W_l denote the activation function, bias, and weight matrix of the hidden layer l, respectively. The output of the MLP hidden layer is defined as:

$$x^{MLP} = a_L\left(\mathrm{W}_L^T\left(a_{L-1}\left(\ldots a_2\left(\mathrm{W}_2^T\begin{bmatrix}\mathrm{p}_u^M \\ \mathrm{q}_i^M\end{bmatrix} + \mathrm{b}_2\right)\ldots\right)\right) + \mathrm{b}_L\right), \quad (3)$$

where p_u^M and q_i^M represent user embedding and item embedding at the MLP layer, respectively. Since ReLU has shown good performance in neural networks [24], we use it as the activation function of the hidden layer. In addition, the ReLU function encourages sparse activation, which is more suitable for sparse data and makes the model less likely to overfit [28].

3) PREDICTIVE LAYER

The prediction layer maps the output of the final hidden layer to the score. Therefore, we will consider mapping the result to the ranking after the MLP model is fused with the MF.

D. FUSION AND LEARNING

Thus far, we have introduced two types of models: linear models represented by MF and nonlinear models represented by MLP. MF fits the linear interaction between users and items while MLP models the nonlinear interaction of latent features to obtain nonlinear features. Inspired by methods such as NeuMF and DeepCF, we use a parallel method to aggregate linear and nonlinear models. To adapt to the difference in embedding size of MF and MLP, we remove the prediction layer of the two models and use the remaining parts as the MF layer and MLP layer of the DNR model. In addition, MF and MLP are allowed to learn embeddings independently. In this way, we obtain the joint representations of user-item pairs corresponding to the linear model and the nonlinear model respectively. Finally, a fully connected layer is used to connect the two layers and assign different weights to the features contained in the joint representation. Since these two models have different advantages and learn prediction vectors from different angles, the connection of the MF layer and the MLP layer will produce more accurate prediction values and more competitive model performance. Figure 3 shows the overall architecture of the DNR model, and the prediction formulation is as follows:

$$\hat{x}_{ui} = \sigma\left(\mathrm{h}^T\begin{bmatrix}x^{MF} \\ x^{MLP}\end{bmatrix}\right), \quad (4)$$

where h and σ denote the edge weight and activation function of the output layer, respectively; x^{MLP} represents the outputs of the MLP layer, is defined as Equation 3; and x^{MF} denotes the outputs of the MF layer, is defined as follows:

$$x^{MF} = \mathbf{p}_u^F \odot \mathbf{q}_i^F, \quad (5)$$

where p_u^F and q_i^F represent user embedding and item embedding at the MF layer, respectively.

It should be noted that the result $\hat{x}_{ui}$ of Equation 6 represents the output of the fusion of MF and MLP, we need to

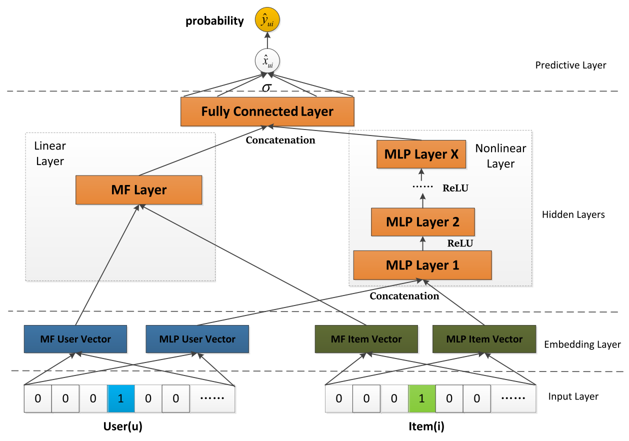

FIGURE 3. **The overall architecture for DNR.**

map it to the probability $\hat{y}_{ui}$ by the softmax function. The probabilities $\hat{y}_{ui}$ that item i ranks first in the list of user u are defined as follows:

$$\hat{y}_{ui} = \frac{e^{\hat{x}_{ui}}}{\sum_{j=1}^{J} e^{\hat{x}_{uj}}}, \quad (6)$$

where J represents the number of items in the list l_u.

Here, we mainly explore the top-N recommendation problem. The probability that items exist in the user preference list is defined as:

$$P(l_u \mid \Theta) = \prod_{i \in l_u^+} \hat{y}_{ui} \prod_{k \in l_u^-} (1 - \hat{y}_{uk}), \quad (7)$$

where Θ represents the model parameter vector; l_u^+ and l_u^- represent the sets of items observed and unobserved by user u in the recommendation list l_u, respectively.

Then, we evaluate the loss of DNR as follows:

$$l(y, \hat{y}) = -\sum_{u=1}^{N}\left(\sum_{i \in l_u^+} \log \hat{y}_{ui} + \sum_{k \in l_u^-} \log(1 - \hat{y}_{uk})\right), \quad (8)$$

where y and $\hat{y}$ represent the true value and the predicted value, respectively; and $l(\cdot)$ denotes a binary cross entropy loss.

Finally, a regularization method is added to the loss function to reduce overfitting. The final objective function is defined as follows:

$$L = l(y, \hat{y}) + \lambda\Omega(\Theta), \quad (9)$$

where $f(\cdot)$ is the loss function of the model; λ is the regularization coefficient set in the experiment; $\Omega(\Theta)$ is the regularization method, which is defined as follows:

$$\Omega(\Theta) = \sum_{l=1}^{L} \| \mathrm{W}_l\|_F^2 + \sum_{u=1}^{N} \|p_u\|_F^2 + \sum_{i=1}^{M} \|q_i\|_F^2, \quad (10)$$

E. PAIRWISE DNR

Pairwise DNR considers the relative ranking of each item and other items and then integrates all the relative order relationships to obtain the overall ranking result.

Compared to listwise DNR, it focuses on the order between two items. By setting the rating as a label, pairwise methods assume that the user prefers the item with a label of 1, rather than 0. In this way, pairwise DNR combines positive and negative instances for training. The objective function is defined as follows:

$$l(y, \hat{y}) = -\sum_{u=1}^{N} \left(\sum_{i \in I_u^+} \log \hat{y}_{ui} + \sum_{k \in I_u^-} \log \left(1 - \hat{y}_{uk}\right) \right), \quad (11)$$

where I_u^+ and I_u^- represent the sets of items with labels 1 and 0 corresponding to the user u, respectively.

Pairwise DNR optimizes ranking loss by evaluating the preference probability between users and each pair of items. We set k = 2 under the framework of the listwise DNR and optimize the objective function to obtain the pairwise DNR model. Figure 4 shows the architecture of pairwise DNR.

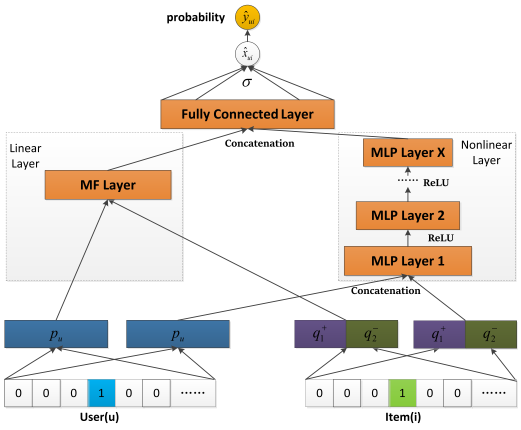

FIGURE 4. **The architecture of pairwise DNR.**

Pairwise DNR only considers the relative position of the items and does not model the accurate position information of the items in the predictive ranking. In addition, pairwise DNR is more sensitive to noise labeling, that is, the wrong order of a pair of items will cause multiple pairs of items to be sorted incorrectly. These defects will affect the training results and limit the sorting performance. However, compared to the list DNR traversing the item list for sorting, the paired DNR greatly reduces the training complexity and improves the model efficiency. All in all, the pairwise DNR will limit part of the model performance but can save a lot of time.This conclusion will be proven in the experimental part.

IV. EXPERIMENT

In this section, we first introduce the experimental settings. Then, we conduct detailed experiments to answer the following questions:

RQ1: How does DNR perform compared with other methods?

RQ2: How do the length of the list and the depth of the model affect DNR respectively?

RQ3: How do we choose between pairwise DNR and listwise DNR in practice?

A. EXPERIMENTAL SETTINGS

1) DATASETS

We evaluated our method on three public data sets: MovieLens100K, Movielens1M, and Yahoo Movies. For all data sets, each user has at least 20 interactions with items and each item has at least 1 interaction with users. The statistics of these three data sets are summarized in Table 1.

TABLE 1. **Statistics of the data sets.**

Statistics	MovieLens100K	Movielens1M	Yahoo Movies
of users	943	6,040	7,642
of items	1,682	3,952	11,915
of ratings	100,000	1,000,000	211,231

2) COMPARISON METHODS

The following briefly introduces some comparison methods to evaluate the performance of the proposed DNR:

BPR [17] focuses on implicit feedback and optimize personalized ranking with a pairwise loss function.

ListRank-MF [19] focuses on the top-one problem in a list, and proposes a method that combines MF with a list-based learning ranking algorithm.

DeepCF [7], which is a point-oriented ranking learning method, combines representation and matching function learning-based CF methods to achieve high ranking performance.

DeepRank [11] focuses on top-n ranking in a list, which sets the embedding of users and items to different sizes and inputs them into MLP for ranking, is one of the state-of-the-art ranking methods.

3) PARAMETER SETTINGS

First, the learning rate and regularization coefficient are set to 0.001 and 10^{-6} in all methods, respectively. Then, the number of latent features in BPR and ListRank-MF is set to 16. For an accurate comparison, the embedding size of DeepCF, DeepRank and our model are also set to 16. Finally, the hidden layer size, number of training iterations, and batch size in DeepCF, DeepRank, and our model are set to 2, 50, and 512, respectively.

4) EVALUATION PROTOCOLS

Follow literature [6], [7], [11], [17], [21], we adopted the *leave-one-out* evaluation method, which uses the latest interaction of each user for testing and utilizes the remaining data for training. However, this method is time-consuming. To speed up the training, instead of re-training the model from random initialization, additional features are added to the user embeddings of the pre-trained model[11]. In addition, for efficiency, we randomly sampled 100 unobserved items for each user as the test data and reduce the ranking list of evaluation metrics to 10. The predicted personalized ranking is evaluated on the test set by the two widely adopted evaluation metrics, *Hit Ratio* (HR) and *Normalized Discounted Cumulative Gain* (NDCG). Intuitively, the HR evaluates the accuracy of the predicted top-10 list, and NDCG highlights the top items by giving higher scores to them.

We performed 10 repeated experiments for each experiment, to discard extreme values and take the average value to avoid data errors caused by model instability.

B. OVERALL PERFORMANCE (RQ1)

We conducted detailed experiments on three data sets and compared the results with comparison methods. To compare the performance of all methods under different numbers of ranking list intuitively, we added the number(n) of top-n items as the abscissa and drawn line graphs as shown in Figure 5.

As shown in Figure 5, in general, the HR and NDCG results are consistent. For all data sets, as the number of top-n items (n) increases, the ranking prediction capabilities of all methods are improved. Among these methods, DNR shows superior ranking performance. On the Movielens1M data set, the performance improvement of DNR compared to other algorithms is more obvious. On the Yahoo Movie data set, the HR and NDCG of the DNR are both reaching the highest values. This shows that our model still has good performance on sparse data sets. In addition, DeeRank ranks the learning methods and captures the user's features from user's pairwise behavior on items, thus making the best performance among the comparison methods. Quantitatively, DNR significantly outperforms the state-of-the-art method DeepRank (on the three data sets of MovieLens100K, Movielens1M, and Yahoo Movies, on average, the improvement over DeepRank is 3.9%, 5.3%, and 1.8%, respectively).

As mentioned before, the BPR and ListRank-MF methods lack complex and deeper interactions in the data. Both DeepRank and DeepCF use the MLP to learn models, which ignore the equally important shallow interactions. Therefore, they show limited performance on all data sets. In addition, the DeepCF method focuses on the pointwise method, which uses user-item pairs to rank items and ignores the pairwise ranking information. To summarize, the DNR method we proposed is more advanced than the comparison methods.

C. IMPACT OF THE LIST LENGTH AND THE MODEL DEPTH (RQ2)

1) IMPACT OF THE LENGTH OF THE LIST

We explored the impact of the list length (K) on listwise DNR with extensive experiments. When K is set to 2, it is equivalent to pairwise DNR. The results are shown in Table 2 and Table 3.

From Table 2 and Table 3, we first find that adding more item information can further improve the performance. On all data sets, evaluation metrics of the model become better with the increase of the value of K. However, the increase in the length of the list will inevitably lead to a substantial increase

TABLE 2. The impact of the length of the list on HR@10.

HR@10		
K	MovieLens100K	MovieLens1M
2	0.7387	0.7377
5	0.7987	0.7997
10	0.8123	0.7993
15	0.8229	0.8154

TABLE 3. The impact of the length of the list on NDCG@10.

NDCG@10		
K	MovieLens100K	MovieLens1M
2	0.4823	0.4958
5	0.5244	0.5328
10	0.5325	0.5425
15	0.5435	0.5639

TABLE 4. The impact of the number of hidden layers on the DNR model.

L	MovieLens100K		MovieLens1M	
	HR@10	NDCG@10	HR@10	NDCG@10
1	0.7922	0.5210	0.7634	0.4922
2	**0.8049**	**0.5244**	**0.7987**	**0.5463**
3	0.7943	0.5167	0.7834	0.5317
4	0.7359	0.4717	0.7767	0.5256
5	0.7434	0.4652	0.7689	0.5139

in the complexity of the model, which greatly increases the model training time. Therefore, we recommend setting K as 5, which saves training time while ensuring the performance.

2) IMPACT OF THE DEPTH OF THE MODEL

Finally, to fully tap the potential of listwise DNR, we tested the model performance with different numbers of the hidden layers. In the experiment, we set the sizes of the hidden layers to [8], [16,8], [32,16,8], [64,32,16,8] and [128,64,32,16,8], and the length of the list to 5.

As shown in Table 4, an appropriate number of hidden layers can improve the performance of the model. When the number of hidden layers is 2, DNR achieves the best results on both data sets. When the number of hidden layers continues to increase, the HR@10 and NDCG@10 of the model begin to decrease. In other words, too many or too few hidden layers will achieve the limited performance of DNR. It is well known that deep neural networks have strong modeling and representation capabilities, but too few layers will lead to insufficient fitting capabilities [11], [25]. Therefore, we propose the following question: Why does the DNR model achieve the optimal effect when there are only two hidden layers? Here, we first show our assumption: the addition of MF enhances the overall performance of DNR, but it will limit the depth of the MLP hidden layers. The following experiment proves our idea.

We first explore the impact of MF fusion on the MLP model under the DNR framework through experiments. When MF is not fused, the performance of the MLP model at different hidden layers is shown in Table 5. Intuitively, HR@10 and NDCG@10 will increase as L increases.

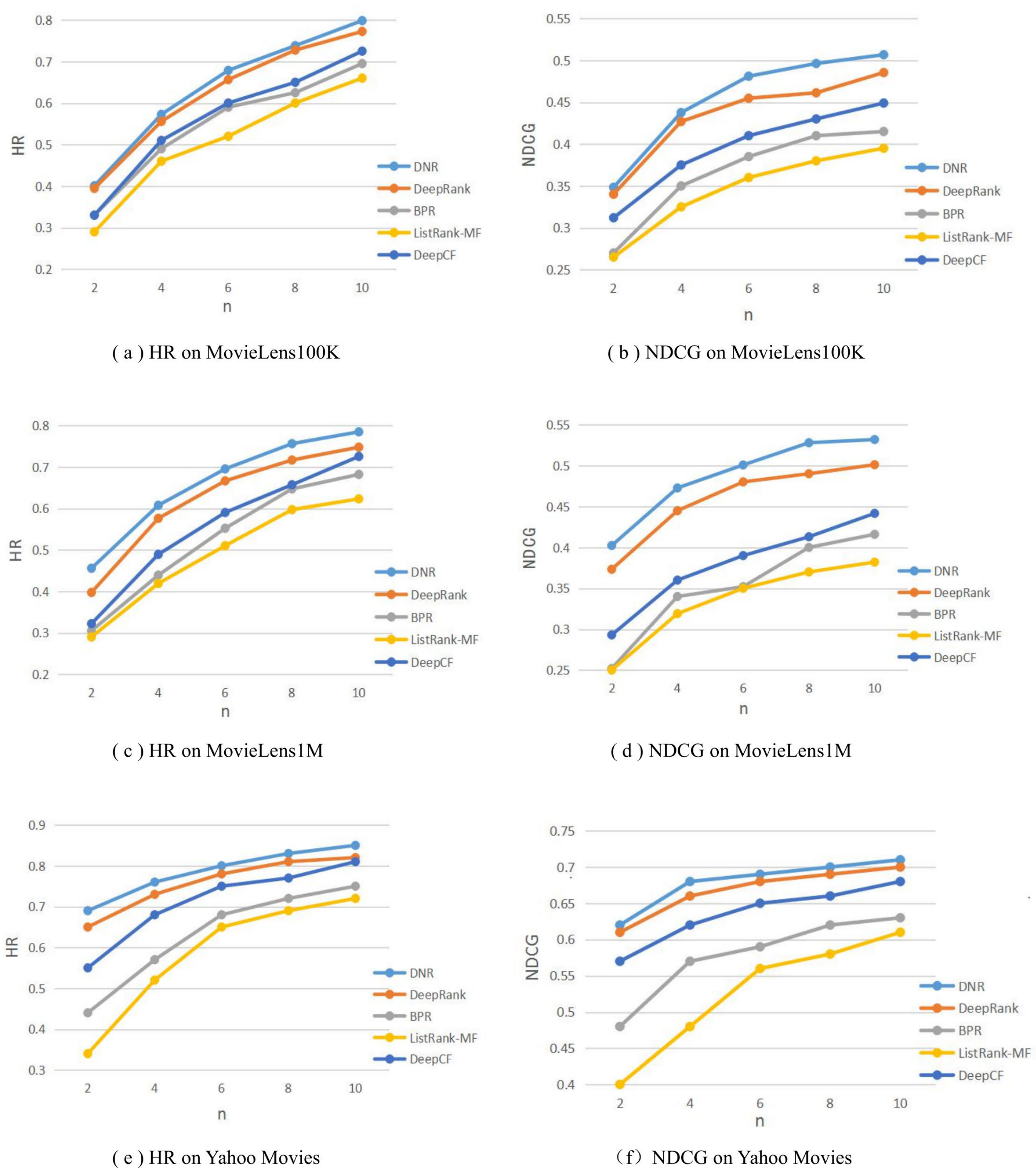

FIGURE 5. Performance comparison under different numbers (n) of top-n items.

When L is set to 4, both indicators reach the maximum. This indicates that the model effect is best when the number of hidden layers of the MLP model is 4. However, in our model, DNR, as shown in Table 4, the two indicators reach the maximum when L is 2 and exceed the optimal indicator of MLP. Obviously, the addition of MF enhances the overall performance of the model but limits the depth of the MLP hidden layers. The reason is that the MLP model outputs a large number of nonlinear features while the MF model only outputs linear features. In theory, the fusion of the two will enhance the performance of the overall model, but this is at the cost of losing certain nonlinear features. When reaching a certain critical point, that is, $L = 2$, the performance of the DNR model is optimal. When the number of hidden layers of

TABLE 5. The impact of the number of hidden layers on the MLP model.

L	MovieLens100K		MovieLens1M	
	HR@10	NDCG@10	HR@10	NDCG@10
1	0.7012	0.4453	0.6919	0.4236
2	0.7370	0.4679	0.7455	0.4666
3	0.7635	0.4892	0.7668	0.5111
4	**0.7738**	**0.4958**	**0.7689**	**0.5207**
5	0.7702	0.4710	0.7628	0.5177

TABLE 6. Performance of pairwise DNR and listwise DNR.

	MovieLens100K		MovieLens1M	
	HR@10	NDCG@10	HR@10	NDCG@10
pairwise DNR	0.7310	0.4623	0.7417	0.4958
listwise DNR	0.8023	0.5244	0.7993	0.5425

TABLE 7. Time cost on pairwise DNR and listwise DNR.

	MovieLens100K	MovieLens1M
pairwise DNR	3 m 20 s	19 m 23 s
listwise DNR	4 m 49 s	35 m 35 s

MLP continues to increase, the noise caused by the loss of nonlinear features will increase, which leads to a decrease in the overall performance of the model. Therefore, the fusion of MF and MLP with an appropriate number of hidden layers can achieve the best results under the DNR framework.

D. COMPARISON OF PAIRWISE DNR MODEL AND LISTWISE DNR MODEL (RQ3)

In this section, we compare the performance and time complexity of the pairwise DNR model and the listwise DNR model through experiments. Table 6 and Table 7 show the performance and the training time of the two models on the MovieLens100K and Movielens1M data sets. Obviously, the performance of the listwise DNR model is far superior to that of the pairwise DNR model. However, the training time of the pairwise DNR model is significantly less than that of the listwise DNR model. On the MovieLens100K data set with a small amount of data, the difference between the two is not obvious, but on the Movielens1M data set with a large amount of data, the pairwise DNR saves nearly half the time as the listwise DNR. The main reason is that the pairwise DNR model has fewer items for one-time training and will take up fewer resources.

In summary, pairwise DNR can be considered if we focus on efficiency. And for higher performance of the list ranking model, listwise DNR is a better choice.

V. CONCLUSION AND FUTURE WORK

In this work, we explored the possibility of fusion based on linear interactions and nonlinear interactions for ranking recommender systems. We designed a framework DNR, which ensembles linear models and nonlinear models, and proposed its instantiation — fusion of MF and MLP. Due to limited space, we will introduce DNR models based on more instantiations in subsequent papers.

In future work, we will complete the following work. First, more instantiation based on the DNR framework will be introduced in subsequent articles. Second, we plan to add some auxiliary information to develop the personalized ranking performance of the model, such as comment information and knowledge graph information. Finally, as the current multi-modal recommendation system is becoming increasingly more popular, we will try to add multi-modal features such as pictures and audio information to the DNR model to develop DNR recommendations for multi-modal content.

REFERENCES

[1] C. He, Q. Zhang, Y. Tang, S. Liu, and H. Liu, "Network embedding using semi-supervised kernel nonnegative matrix factorization," *IEEE Access*, vol. 7, pp. 92732–92744, 2019, doi: 10.1109/ACCESS.2019.2927496.

[2] H. Dai, L. Wang, and J. Qin, "Metric factorization with item cooccurrence for recommendation," *Symmetry*, vol. 12, no. 4, p. 512, 2020, doi: 10.3390/sym12040512.

[3] S. Wang, S. Huang, T. Liu, and J. Ma, "Ranking-oriented collaborative filtering: A listwise approach," *ACM Trans. Inf. Syst.*, vol. 35, no. 2, pp. 1–28, 2016, doi: 10.1145/2960408.

[4] H. Li, "Internet tourism resource retrieval using PageRank search ranking algorithm," *Complexity*, vol. 2021, pp. 1–11, May 2021, doi: 10.1155/2021/5114802.

[5] M. S. Pera and Y.-K. Ng, "A personalized recommendation system on scholarly publications," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2011, pp. 2133–2136, doi: 10.1145/2063576.2063908.

[6] Q. Zhang, L. Cao, C. Zhu, and Z. Li, "Learning explicit and implicit user-item couplings in recommendation for deep collaborative filtering," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 3662–3668, doi: 10.24963/ijcai.2018/509.

[7] Z. Deng, L. Huang, and C. Wang, "DeepCF: A unified framework of representation learning and matching function learning in recommender system," in *Proc. AAAI*, 2019, pp. 61–68, doi: 10.1609/aaai.v33i01.330161.

[8] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-N recommender systems," in *Proc. 9th ACM Int. Conf. Web Search Data Mining*, Feb. 2016, pp. 153–162, doi: 10.1145/2835776.2835837.

[9] L. Blédaité and F. Ricci, "Pairwise preferences elicitation and exploitation for conversational collaborative filtering," in *Proc. 26th ACM Conf. Hypertext Social Media (HT)*, 2015, pp. 231–236, doi: 10.1145/2700171.2791049.

[10] S. Qiu, J. Cheng, T. Yuan, C. Leng, and H. Lu, "Item group based pairwise preference learning for personalized ranking," in *Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2014, pp. 1219–1222, doi: 10.1145/2600428.2609549.

[11] M. Chen and X. Zhou, "DeepRank: Learning to rank with neural networks for recommendation," *Knowl.-Based Syst.*, vol. 209, Dec. 2020, Art. no. 106478, doi: 10.1016/j.knosys.2020.106478.

[12] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: A factorization-machine based neural network for CTR prediction," 2017, *arXiv:1703.04247*.

[13] C.-D. Wang, Z.-H. Deng, J.-H. Lai, and P. S. Yu, "Serendipitous recommendation in E-commerce using innovator-based collaborative filtering," *IEEE Trans. Cybern.*, vol. 49, no. 7, pp. 2678–2692, Jul. 2019, doi: 10.1109/TCYB.2018.2841924.

[14] Z. Zhao, L. Huang, C. Wang, and D. Huang, "Low-rank and sparse cross-domain recommendation algorithm," in *Proc. DASFAA*, 2018, pp. 150–157, doi: 10.1007/978-3-319-91452-7_10.

[15] Q.-Y. Hu, Z.-L. Zhao, C.-D. Wang, and J.-H. Lai, "An item orientated recommendation algorithm from the multi-view perspective," *Neurocomputing*, vol. 269, pp. 261–272, Dec. 2017, doi: 10.1016/j.neucom.2016.12.102.

[16] Z. Zheng, K. Chen, G. Sun, and H. Zha, "A regression framework for learning ranking functions using relative relevance judgments," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2007, pp. 287–294.

[17] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. 25th Conf. Uncertainty Artif. Intell. (UAI)*, 2009, pp. 452–461.

[18] S.-T. Park and W. Chu, "Pairwise preference regression for cold-start recommendation," in *Proc. 3rd ACM Conf. Recommender Syst. (RecSys)*, 2009, pp. 21–28, doi: 10.1145/1639714.1639720.
[19] Y. Shi, M. Larson, and A. Hanjalic, "List-wise learning to rank with matrix factorization for collaborative filtering," in *Proc. 4th ACM Conf. Recommender Syst. (RecSys)*, 2010, pp. 269–272, doi: 10.1145/1864708.1864764.
[20] H.-T. Cheng, "Wide & deep learning for recommender systems," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, 2016, pp. 7–10. [Online]. Available: https://arxiv.org/pdf/1606.07792.
[21] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web (WWW)*, 2017, pp. 173–182, doi: 10.1145/3038912.3052569.
[22] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proc. 22nd Int. Conf. Mach. Learn. (ICML)*, 2005, pp. 89–96, doi: 10.1145/1102351.1102363.
[23] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Comput.*, vol. 42, no. 8, pp. 30–37, Aug. 2009, doi: 10.1109/MC.2009.263.
[24] X. He, Z. He, J. Song, Z. Liu, Y.-G. Jiang, and T.-S. Chua, "NAIS: Neural attentive item similarity model for recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 12, pp. 2354–2366, Dec. 2018, doi: 10.1109/TKDE.2018.2831682.
[25] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 3203–3209, doi: 10.24963/ijcai.2017/447.
[26] J. Gong, W. Du, H. Li, Q. Li, Y. Zhao, K. Yang, and Y. Wang, "Score prediction algorithm combining deep learning and matrix factorization in sensor cloud systems," *IEEE Access*, vol. 9, pp. 47753–47766, 2021, doi: 10.1109/ACCESS.2020.3035162.
[27] D. Zhou, S. Hao, H. Zhang, C. Dai, Y. An, Z. Ji, and I. Ganchev, "Novel SDDM rating prediction models for recommendation systems," *IEEE Access*, vol. 9, pp. 101197–101206, 2021, doi: 10.1109/ACCESS.2021.3097207.
[28] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Stat. (AISTATS)*, vol. 15, 2011, pp. 315–323.
[29] M. Chen, Y. Li, and X. Zhou, "CoNet: Co-occurrence neural networks for recommendation," *Future Gener. Comput. Syst.*, vol. 124, pp. 308–314, Nov. 2021, doi: 10.1016/j.future.2021.06.008.
[30] R. Cai, J. Wu, A. San, C. Wang, and H. Wang, "Category-aware collaborative sequential recommendation," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2021, pp. 388–397, doi: 10.1145/3404835.3462832.

CHUNTING WEI received the bachelor's degree from Guangdong Ocean University, in 2018. He is currently pursuing the master's degree with the College of Information Science and Engineering, Xinjiang University. His research interest includes recommender systems.

JIWEI QIN received the master's and Ph.D. degrees from the School of Computer Architecture, Xi'an Jiaotong University, Xi'an, China, in 2008 and 2013, respectively. She is currently a Professor with Xinjiang University. Her research interests include data mining and recommender systems.

WEI ZENG received the bachelor's degree from the Jiangxi University of Science and Technology, in 2019. He is currently pursuing the master's degree with the College of Information Science and Engineering, Xinjiang University. His research interest includes recommender systems.