Expert Systems With Applications 195 (2022) 116595

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

CoCNN: Co-occurrence CNN for recommendation

Ming Chen, Tianyi Ma, Xiuze Zhou [*]

Hithink RoyalFlush Information Network Co., Ltd., Hangzhou 310023, China

ARTICLE INFO

Keywords:
Collaborative Filtering
CNN
Deep learning
Neural network
Co-occurrence pattern
Multi-task

ABSTRACT

Under the assumption that items are independent and identically distributed, most existing CF methods learn representation from user–item pairs, but ignore the connections among items, leading to limited performance. Considering the challenge of recommendation, we propose a novel neural network, CoCNN, which combines a Co-occurrence pattern and CNN for CF with implicit feedback. The key idea of the co-occurrence pattern is that some items always appear between pairs on a user's favorite list. In CoCNN, co-occurrence relationships act as a bridge in user–item pairs and item–item pairs, which are not observed directly. To model user–item and item–item information simultaneously, we propose a multi-task neural network to share the knowledge of the two tasks. Finally, experimental results demonstrate that CoCNN successfully captures more useful information, and therefore can be used as a simple and effective tool for recommendation. Our projects are available online at https://github.com/XiuzeZhou/CoCNN.

1. Introduction

Facing a huge number of choices, consumers find it difficult to locate quickly what they want (Chen & Zhou, 2020; Liu et al., 2021; Wang, Gao, Peng, & Mo, 2019). To alleviate this problem, recommender systems are designed to provide a list of items for users (Choi, Jeong, Lee, & Lee, 2021; Costa & Dolog, 2019; Du, Tang, & Ding, 2019; Zhang, Yao, Sun, & Tay, 2019). Recommender systems are being successfully used in daily applications, such as e-commerce (Cai, Filos-Ratsikas, Tang, & Zhang, 2018; Zhao et al., 2020), news (Hu, Li, Shi, Yang, & Shao, 2020; Wu, Wu, et al., 2019), music (Schedl, Knees, McFee, Bogdanov, & Kaminskas, 2015; Van Den Oord, Dieleman, & Schrauwen, 2013), and education (Benhamdi, Babouri, & Chiky, 2017; Lin et al., 2021).

In recommender systems, Collaborative Filtering (CF), one of the most popular and widely used techniques, learns users' interests from their historical interactions with items (Chen, Xin, Wang, & Ding, 2021; Li, Chen, Lu, Chu, & Gu, 2019). CF assumes that similar users are interested in similar items (Ebesu, Shen, & Fang, 2018; Zhou & Wu, 2016). Based on this assumption, various models have been proposed; among them, Matrix Factorization (MF) is the most successful and famous (Koren, Bell, & Volinsky, 2009; Shen et al., 2019).

Recent advances in deep learning in Natural Language Processing (NLP) (Al-Ayyoub, Nuseir, Alsmearat, Jararweh, & Gupta, 2018; Mou et al., 2016), Computer Vision (CV) (Hongtao & Qinchuan, 2016; Khan, Rahmani, Shah, & Bennamoun, 2018), and medical applications (Shen, Wu, & Suk, 2017; Wu, Zeng, Lin, & Zhou, 2021) have inspired scientists to explore this application for recommender systems (Covington, Adams, & Sargin, 2016; Van Den Oord et al., 2013; Zhang, Yao, Sun, & Tay, 2019). The major advantage of deep learning methods is they have the powerful ability to learn deep and non-linear representation from raw data (Cheng et al., 2016; LeCun, Bengio, & Hinton, 2015; Wang, Wu, Lou, & Jiang, 2022; Zhu, Wu, Qiang, Yuan, & Li, 2021). For example, Multi-Layer Perceptron (MLP), which extracts useful high-order features from its hidden layers, is used very successfully to learn user preference (He et al., 2017; Lu et al., 2018). Long Short-Term Memory (LSTM), with the strong ability to capture temporal information from a sequence, is designed to simulate user behavior to predict tendencies (Zhou, Huang, Hu, Zhu, & Tang, 2018; Zhu et al., 2017). Convolutional Neural Network (CNN), which has demonstrated its power to represent abstract features and make significant achievement in image processing, is developed to learn high-order correlations for better recommendations (He et al., 2018); Autoencoder, a deep neural network architecture that aims to learn the lower-dimensional representation of raw data and reconstruct the data from this representation, is also successfully applied in recommender systems (Sedhain, Menon, Sanner, & Xie, 2015). These deep learning models greatly boost the development of recommender systems.

However, the common problem with most existing CF methods is they assume items are independently and identically distributed and neglect the various coupling relationships between items, leading to limited performance (Chen, Li, & Zhou, 2021; Sun et al., 2017; Wu, Tang, et al., 2019). To solve this problem, (Liang, Altosaar, Charlin, & Blei, 2016) assuming two frequently consumed items are similar, proposed a model CoFactor that combines MF with the item

[*] Corresponding author.
E-mail addresses: chm@zju.edu.com (M. Chen), matianyi@myhexin.com (T. Ma), zhouxiuze@foxmail.com (X. Zhou).

https://doi.org/10.1016/j.eswa.2022.116595
Received 20 November 2021; Received in revised form 4 January 2022; Accepted 19 January 2022
Available online 5 February 2022

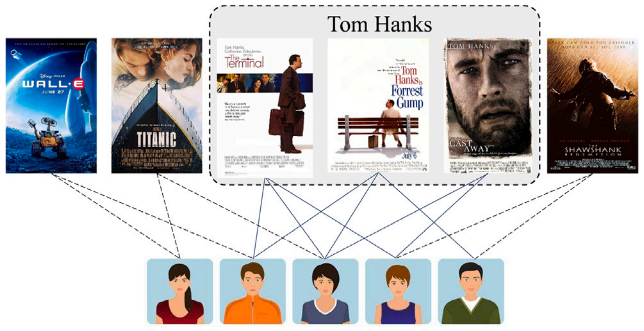


Fig. 1. A toy-example of co-occurrence patterns in movie.

co-occurrence matrix calculated by Shifted Positive Pointwise Mutual Information (SPPMI). In CoFactor, the co-occurrence pattern of the items in the feedback data improves MF performance. To enhance the recommendation, Wu, Zhou, Nie, and Cao (2020) developed Co-occurrence embedding Regularized Metric Learning model (CRML), which integrated co-occurrence information in metric learning and used co-occurrence embedding for regularization. By making full use of the co-occurrence information in the interactions, CRML better depicts the relationships between items and achieves better recommendation performances. Also, to obtain highly descriptive features, Chen, Li, and Zhou (2021) designed a neural network on the co-occurrence pattern.

A toy-example of the movie co-occurrence pattern is illustrated in Fig. 1. When one user (who is a fan of Tom Hanks) watches the famous movies of Tom Hanks, such as "Forrest Gump", "The Terminal", and "Cast Away", he/she is very likely to give them a high rating. Thus, to capture the relationship between items, item–item co-occurrence information is used to build our model.

For the co-occurrence modeling of our model, we also assume that the more frequently two items appear together, the closer they are to each other. Take Fig. 1 for example, in users' records, Tom Hanks' films, such as "Forrest Gump", "The Terminal", and "Cast Away", always appear together. Thus, we argue that all co-occurrence films are closely related. To better discover the relationship between items, our models train on co-occurrence patterns.

Inspired by the item co-occurrence interest mode, we propose a novel model: Co-occurrence pattern combined with CNN (CoCNN) for CF with implicit feedback. We assume that two highly correlated items, co-liked by users who have similar states, are close to each other. Thus, the co-occurrence pattern is used to explore the relationships among items. Instead of calculating the item co-occurrence matrix by SPPMI, we reconstruct raw interactions to the co-occurrence matrix, whose element is one when the user interacts with two different items.

Then, to learn high-order correlations between embeddings, He et al. (2018) proposed a CNN-based architecture, Outer product-based Neural Collaborative Filtering (ONCF), which builds the user–item interactions by outer product. However, in recommendation tasks, ONCF suffers from two main challenges: (1) ONCF focuses only on the interaction between a single user–item pair and ignores the relationships between items; (2) Outer product operation increases the data exponentially, greatly increasing space complexity. To address both challenges, we designed a CNN architecture to learn representations from the co-occurrence matrix and explore the parameter sharing strategy. In CoCNN, CNN is applied directly to the embedding, rather than through intermediate steps by outer product operations.

The main contributions are summarized as follows:

(1) We design a novel neural architecture, which captures user–item and item–item relationships across all user interactions through the co-occurrence pattern, a simple and effective structure to operate directly on the pairwise items of the user.

(2) We design a CNN-based framework in which a filter with size 2×2 catches the link between a user and an item, and a filter with size 3×3 catches the links between a user and his items to enhance recommendation performance;

(3) We provide a new insight into the recommendation tasks, including the designing of two filters and the co-occurrence pattern. Our model is easily applicable because of its simplicity and effectiveness.

2. Related work

We start with formulating the problem to be solved, and then introduce some methods of item co-occurrence in recommendations. Finally, we give a brief review of a widely applied neural-based frameworks in recommendations: ONCF, a CNN-based framework for CF.

2.1. Problem definition

In recommendation tasks, given two sets: U with m users, I with n items, their interactions are denoted by $R \in \mathbb{R}^{m\times n}$. In R, the element, $r_{ui} = 1$ denotes an observed interaction between u and i; otherwise $r_{ui} = 0$.

Then, a matrix of the user's pairwise preference, $Y \in \mathbb{R}^{m\times n\times n}$, is generated from R. In Y, the element, $y_{uij} = 1$ denotes that for user, u, items i and j are co-occurring; $y_{uij} = 0$ denotes that items i and j never appeared together.

$$y_{uij} = \begin{cases} 1, & \text{if } r_{ui} = 1 \text{ and } r_{uj} = 1 \\ 0, & \text{otherwise} \end{cases}. \tag{1}$$

2.2. Methods for item co-occurrence

In recommendation tasks, MF is dominant in CF among various techniques. Most existing methods train user–item interactions independently, separating examples into distinct and unrelated instances, which ignores the relationships between items. To solve this problem, Liang et al. (2016) propose a method, CoFactor, with an item co-occurrence, which assumes that frequently co-occurring items are likely to be rated by similar users. By adding the items co-occurrence information in MF, CoFactor achieves a high level of recommendation performance. Also, to improve recommendation quality, Wu et al. (2020) propose CRML, which combined co-occurrence information in user–item interactions, to better depict the relationships between items and achieve better performances. Inspired by the success of the items co-occurrence pattern, to better learn the relationships between items, we designed a novel neural network, CoCNN.

2.3. ONCF

ONCF is a CNN-based framework for CF (He et al., 2018). In ONCF, the outer product is used to build a matrix about the pairwise correlations of users. Then, the matrix is fed to deep CNN by filters, just like image processing. Neural networks are a great tool to learn nonlinear features from data (Chen, Xin, Wang, & Ding, 2021; Choi et al., 2021; Wang, Wang, Shi, Song, & Li, 2020). The framework of ONCF is shown in Fig. 2.

Then, the Bayesian Personalized Ranking (BPR) objective, which assumes that observed items rank higher than unobserved ones, is used to learn user pairwise preferences by considering the relative importance of one item in the pairwise pattern (Rendle, Freudenthaler, Gantner, & Schmidtthieme, 2012; Wang et al., 2016). L2 regularization is used to prevent over-fitting. The objective is defined as follows:

$$\mathcal{L}_o = -\sum_{u} \sum_{i\in O_u \cap j\in O_u^-} \ln \sigma\left(\hat{r}_{ui} - \hat{r}_{uj}\right) + \lambda \|\Theta\|_F^2 , \tag{2}$$

where O_u and O_u^- denote the set of positive and negative instances of u, respectively; and Θ is the learning parameter set of the model.

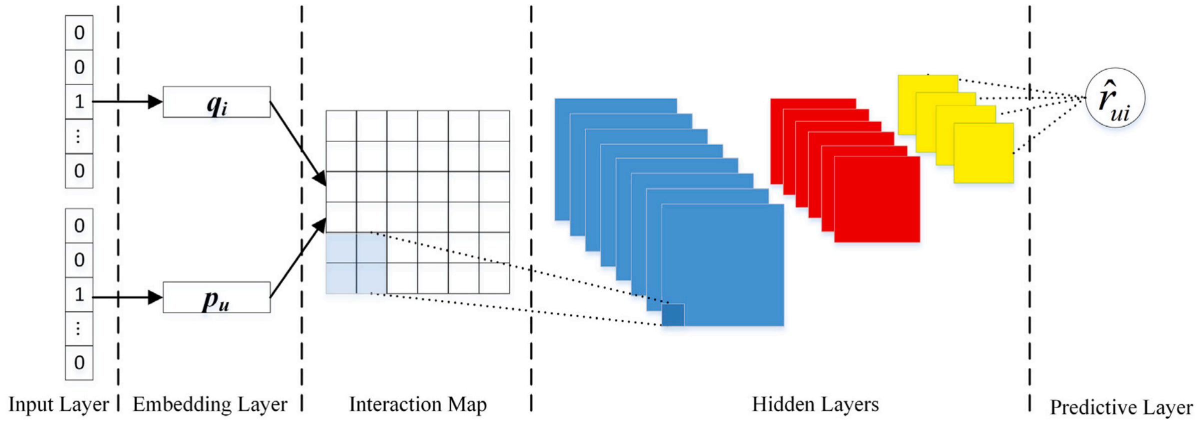


Fig. 2. Framework of ONCF.

Table 1
Comparisons of different recommendation methods.

Methods	Deep model	Low cost	Item relationship
MF	×	√	×
CoFactor	×	√	√
CRML	×	×	√
ONCF	√	×	×
CoCNN	√	√	√

Although OCNF performs well in recommendation tasks, it faces two problems: (1) It is modeled on user–item pairs and neglects the latent relationship between items; (2) Outer product operations increase raw data exponentially, greatly exacerbating the computational burden. Table 1 shows some comparisons of different recommendation methods.

3. The proposed method

3.1. CoCNN

First, we assume that users co-rate similar items in pairs. Next, we propose our models and design their frameworks. Finally, we describe the learning optimization of our models. Our proposed methods aim to learn personalized ranking recommendations from user–item interaction information. There are two key designs in our models: a filter with size 2 × 2 catches the link between a user and the item, and a filter with size 3 × 3 catches the links between a user and his items.

Through several measures, such as local receptive fields, shared weights and sampling, CNN effectively captures local features of images or documents (Kim, Park, Oh, Lee, & Yu, 2016). To effectively capture the relationship of the user–item pair with CNN, we designed a new architecture. To generate a matrix for the filter of CNN, instead of using an outer product like ONCF, we connect embeddings directly. In CoCNN, the embeddings of user u and his pairwise preference for items i and j are combined into a matrix. Then, the matrix is fed to CNN to learn pairwise behavior. To model the relationships of user–item and item–item, we designed two ways, and their frameworks are shown in Fig. 3.

Input and Embedding Layers. Like most neural network models of recommendation (Chen, Li, & Zhou, 2021; Chen & Zhou, 2020; Cheng et al., 2016; Costa & Dolog, 2019; Du et al., 2019; Ebesu et al., 2018; He et al., 2018, 2017; Lin et al., 2021; Shen et al., 2019; Wu, Tang, et al., 2019; Zhang, Yao, Sun, & Tay, 2019), one-hot representations are used to represent each user and item, whose sparse features are mapped from input to dense representation by the embedding layer. The embeddings, $p_u \in \mathbb{R}^{1\times k}$ and $q_i \in \mathbb{R}^{1\times k}$, generated from lookup-tables, are defined as follows:

$$p_u = f_{lookup}(u), \tag{3}$$

$$q_i = f_{lookup}(i). \tag{4}$$

CNN Layers. This layer contains convolution, pooling layers. CNN is powerful to learn deep features from the matrix. The connection matrix is fed to the CNN layer to learn the non-linear correlations between the different embeddings.

Connection. To learn the relationships of user–item and item–item, a new neural network was designed. Given a triple $\left(p_u, q_i, q_j\right)$, a connection matrix integrating them is concatenated as: $c = \left[q_i; p_u; q_j\right] \in \mathbb{R}^{3\times k}$. Then, to connect the embeddings before feeding to the CNN layer, we designed two different ways.

3 × 3 Filter. In CoCNN with 3 × 3 filter (CoCNN3), a full receptive field provides our model with overall relationships between users and their co-occurrence items. A larger receptive field tends to have a more powerful ability to capture more information from input data (Song et al., 2017; Yu et al., 2018).

$$h = ReLU\left(W_3 * c + b_3\right), \tag{5}$$

where $*$ denotes convolution operation; and W_3 denotes a filter of 3×3 size in convolution operation, and the stride is 1.

2 × 2 Filter. In CoCNN with 2 × 2 filter (CoCNN2), the differences from the framework of CoCNN3 are the connection way, the filter size for the connection matrix, and the depth of CNN layers. Compared with CoCNN3, CoCNN2 first uses a filter with a smaller 2×2 size to capture the user–item correlations:

$$h' = ReLU\left(W_2' * c + b_2'\right), \tag{6}$$

where W_2' denotes a filter of 2 × 2 size in convolution operation, and the stride is 1.

Then in the deeper layer, a filter with 2×2 size continues to be used to learn the correlation between users and their co-occurring items. CoCNN2 learns deeper features by its deep CNN structure as follows:

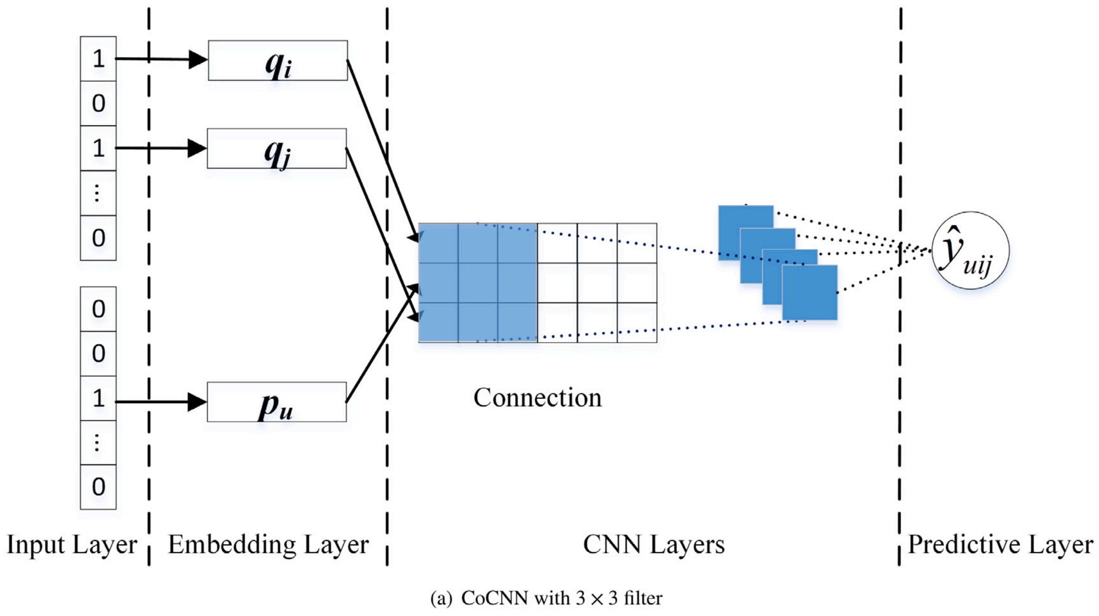
$$h = ReLU\left(W_2 * h' + b_2\right), \tag{7}$$

where W_2 denotes a 2 × 2 filter, and the stride is 1.

Predictive Layer. Finally, to predict probability, $\hat{y}_{uij}$, a fully connected layer is used to map the outcome of the CNN layer to the result, formulated as follows:

$$\hat{y}_{uij} = \sigma\left(W_p^T h + b_p\right), \tag{8}$$

where W_p and b_p denote weight and bias of the predictive layer, respectively; $\sigma(\cdot)$ denotes the *sigmoid* function.



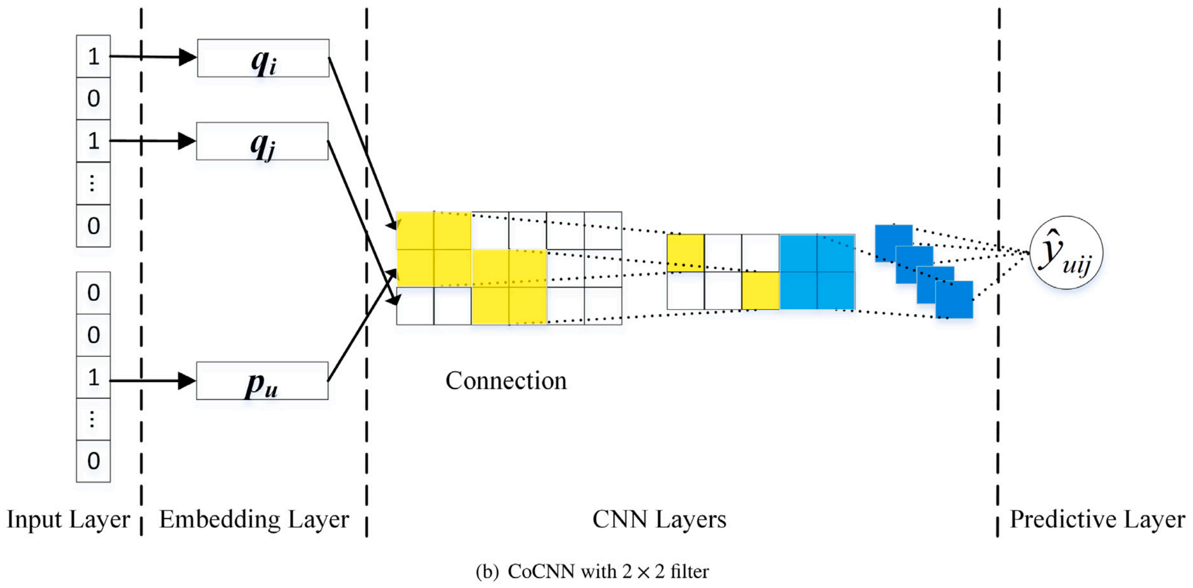


Fig. 3. Two frameworks of CoCNN.

In this task, we aim to learn users' interests from their relative preference for different item pairs. The Mean Squared Error (MSE) function is used to evaluate loss, defined as follows:

$$\mathcal{L}_{co} = -\sum_{u=1}^{n} \sum_{(i,j)\in C\cup C^-} \left(y_{uij} - \hat{y}_{uij}\right)^2, \tag{9}$$

where C_u and C_u^- denote the set of positive and negative instances of u, respectively.

3.2. CNN for user–item

The framework of the CNN for user–item pairs is shown in Fig. 4. In the framework, the first two layers (input and embedding) are the same as in the previous frameworks. The key difference is in the connection part. In the connection of CNN, we combine two embeddings of a user and his item. The remaining layers (CNN and predictive) are the same as the framework for the CoCNN.

In this task, the goal is to learn users' interests from their interactions. Similarly, the MSE function is also used to measure the loss:

$$\mathcal{L}_{ui} = -\sum_{(u,i)\in O\cup O^-} \left(r_{ui} - \hat{r}_{ui}\right)^2. \tag{10}$$

Finally, for better performance, we jointly optimize two tasks in a unified way:

$$\mathcal{L} = \mathcal{L}_{co} + \alpha \mathcal{L}_{ui}, \tag{11}$$

where α is the hyperparameter used to tune the impact between the accuracy and the co-occurrence loss.

Finally, after the network is well trained, to predict the possibility, $\hat{y}_{ui}$, we set $q_i = q_j$, i.e. (u, i, i) as the input to make predictions.

4. Experiments

Our proposed models aim to learn personalized strategy from user–item interactions by co-occurrence neural networks. There are two key

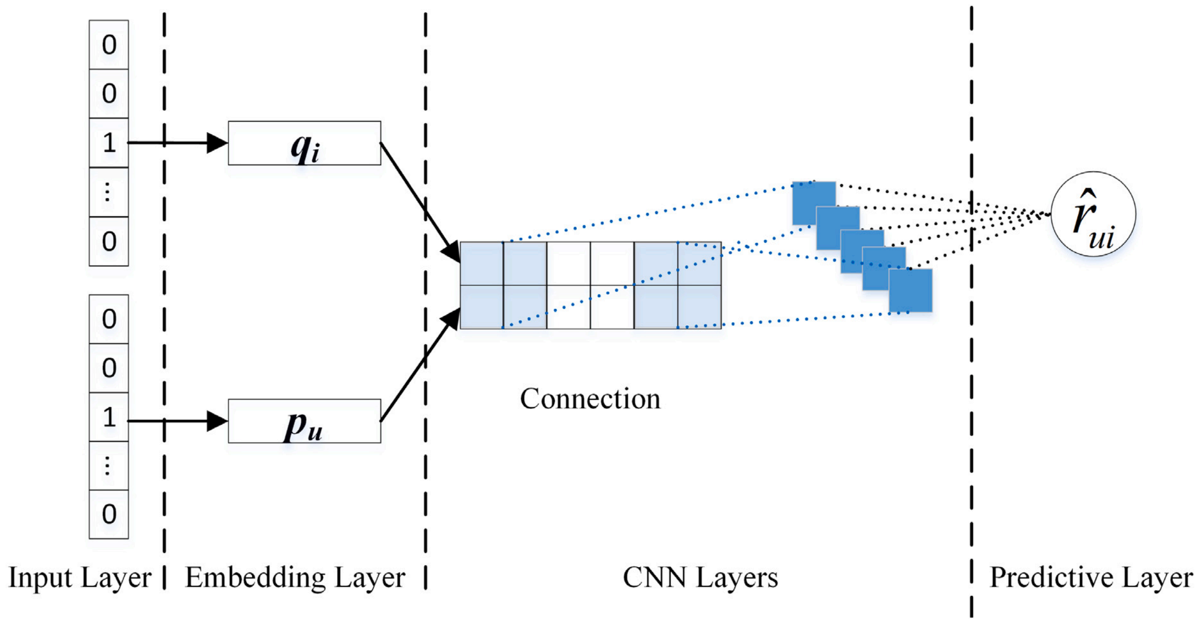


Fig. 4. Framework of CNN for user–item.

Table 2
Statistics for all data sets.

	# of Users	# of Items	# of Interactions	Density
MovieLens100K	943	1682	100,000	6.30%
MovieLens1M	6040	3706	1,000,209	4.47%
Lastfm	518	3488	46,172	0.26%

designs in our models: (1) co-occurrence pattern, which aims to capture the relationships of user–item and item–item; (2) CNN, which aims to capture hidden features. To illustrate the effectiveness of those two components, we conducted experiments on some publicly available data sets.

4.1. Experimental setting

Data sets and Preprocessing. Three data sets: MovieLens100K, MovieLens1M, and Lastfm. The first two, collected by GroupLens (Harper & Konstan, 2015), are available from its web site.[1] The last one, collected from Last.fm, is available from the GroupLens web site.[2] Some statistics are shown in Table 2.

Regarding the leave-one-out evaluation: to evaluate our models, for each user, the latest rated item is selected for testing, and the remainder is used for training. Similar to He et al. (2017), Rendle et al. (2012), Strub and Mary (2015) and Xue, Dai, Zhang, Huang, and Chen (2017), for test data, we randomly sample one item from a user's interactions and 99 items that he never interacts with. Interactions are chosen according to two rules: (1) items with more than five records; (2) users with more than twenty records (Chen, Li, & Zhou, 2021; He et al., 2018, 2017; Wang et al., 2019).

Evaluation Metrics. In recommendation tasks, Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) are popular to assess the capability of recommendation models. The former focuses on prediction accuracy; the latter focuses on the items at the top of the recommendation list.

HR of the Top-K items ($HR@10$) means the percentage of items recommended to users that are successful. If hit_u indicates the number of the right items in the list, HR is defined as follows:

$$HR@K = \frac{\sum_{u=1}^{m} hit_u}{m \times K}.$$

$NDCG@10$ means the predicted position of recommendation lists for all users:

$$NDCG@K = \frac{DCG@K}{IDCG@K}$$

$$DCG@K = \sum_{i=1}^{K} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

where IDCG@K denotes the Top-K list of the best recommendation results for a user; rel_i denotes the graded relevance of the item ranked at position i in the list of recommendations.

Baseline Approaches. We compared our models with the following methods:

- ItemPop (Sarwar, Karypis, Konstan, & Riedl, 2001), a non-personalized model, ranks items by the number of interactions;
- BPR (Rendle et al., 2012), a pairwise ranking method to learn users' preference on two items from the one class data in recommendation tasks;
- CoFactor (Liang et al., 2016), a MF-based method, adds item co-occurrence regularization into MF to improve recommendation quality;
- NeuMF (He et al., 2017), a neural network, combines MF and MLP to learn user interests from implicit feedback;
- ONCF (He et al., 2018), a CNN-based network, uses outer product to generate a matrix followed by CNN layers to provide recommendations.

Parameter settings. Our models have four key parameters: embedding size (d), learning rate (τ), one regularization parameter for learning parameters (λ), and the ratio of each task (α).

To compare the neural frameworks, it is more meaningful to fix the dimension size of the embedding (Rendle, Krichene, Zhang, & Anderson, 2020). As a result, in all models, some key parameters are set

1 https://grouplens.org/datasets/movielens/
2 https://grouplens.org/datasets/hetrec-2011/

Table 3
$HR@10$ and $NDCG@10$ scores of all models.

		ItemPop	BPR	CoFactor	NeuMF	ONCF	CoCNN
MovieLens100K	HR	0.4163	0.6801	0.6878	0.6886	0.6991	0.7093
	NDCG	0.2407	0.3949	0.4013	0.4008	0.4066	0.4142
MovieLens1M	HR	0.4638	0.6932	0.7009	0.7025	0.7030	0.7063
	NDCG	0.2694	0.4123	0.4167	0.4201	0.4220	0.4309
Lastfm	HR	0.4822	0.7044	0.7082	0.7111	0.7204	0.7508
	NDCG	0.3378	0.4923	0.5135	0.5096	0.5133	0.5301

to the same value: d, τ, and λ are set at 32, 0.01, and 10^{-6}, respectively; for all neural frameworks, Adam optimizer is chosen to optimize them, and epochs and batch size are set at 32 and 1024, respectively; for our models, α is set at 1.

4.2. Results and analysis

4.2.1. Overall performance comparison

First, some experiments were performed to verify the performance of our methods on different data sets. The HR and NDCG results are shown in Table 3. For both metrics, our models perform best.

In addition, the following can be observed from the results shown in Table 2: (1) Our methods consistently outperform other baselines in all cases, which shows that in recommendation tasks, our model has a strong ability to make predictions. (2) Among the baseline methods, BPR and CoFactor perform worse than our model, because their nature is linear and their expressive ability is limited when compared with non-linear approaches. (3) In terms of average HR scores, CoCNN outperforms ONCF, the best baseline, by about 1.5%, 0.5%, and 4.2% on MovieLens100K, MovieLens1M, and Lastfm data sets, respectively. In terms of average NDCG scores, CoCNN outperforms ONCF by about 1.9%, 2.1%, and 3.3% on MovieLens100K, MovieLens1M, and Lastfm data sets, respectively. This shows that, compared with our method that directly connects user and item features for CNN, outer product is unnecessary. Outer product easily causes an over-fitting problem when input data is small; (4) NeuMF and ONCF perform worse than our models. The potential reason is that both methods are modeled on user–item pairs and fail to consider the effects of the relationships among items. For better performance, user–item correlations may be insufficient, but jointly learning of user–item and item–item correlations extracts more useful features.

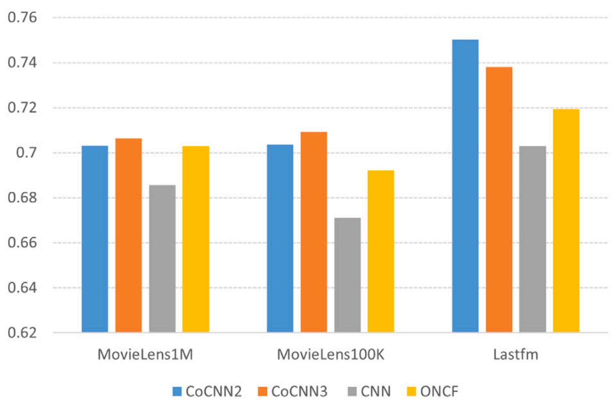
4.2.2. Effect of co-occurrence pattern

Then, to fully evaluate the performance of the co-occurrence pattern in our proposed models, we performed some experiments to compare our models with their simplified versions without the co-occurrence pattern. (See CNN for user–item pairs, and the ONCF for user–item pairs, and the results shown in Fig. 5). In these experiments, we set embedding size, d, and learning rate, τ, at 32 and 0.01, respectively.

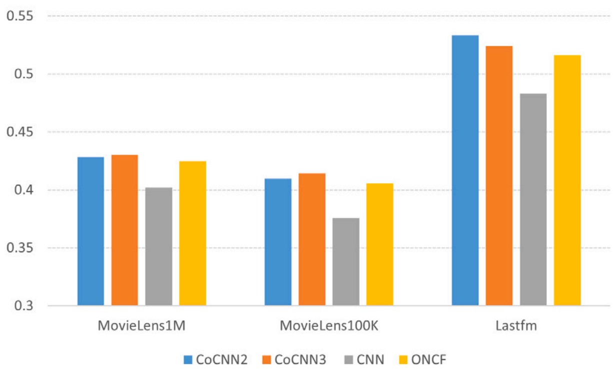
From Fig. 5, we observe the following: First, compared with the results from CoCNN2 and CoCNN3 on the same data set, the results from their simplified methods and ONCF are worse, which shows that a co-occurrence pattern develops CF methods of user–item pairs for better performance, because more detailed information and features can be extracted by a co-occurrence pattern.

Second, on the same data set, compared with CoCNN2, CoCNN3 achieved comparative results. The most probable reason is that both co-occurrence patterns applied in our model can automatically learn the relationships among items, thereby contributing to the development of recommendation quality.

Third, compared with ONCF, which uses outer product to connect the embeddings, our model, CNN, which directly combines the embeddings, also achieves comparative results. Thus, feeding embeddings directly to CNN is also effective.



(a) HR@10 scores



(b) NDCG@10 scores

Fig. 5. Effect of the co-pattern on all data sets.

Finally, comparing the results of the CoCNN models with CNN, we observe that on a large data set, MovieLens1M, CoCNN3 performs better. On the MovieLens100K data sets, CoCNN2 and CoCNN3 achieve comparative results. On the smallest data set (Lastfm), CoCNN3 performs slightly worse than CoCNN2. The potential reason is that the deeper layer of CoCNN2 learns more deep features from the data.

4.2.3. Effect of dimension size

Third, to further examine the effect of dimension size (which represents the size of the information carried by the embedding), on the performance of our model, we chose d from $\{8, 16, 32, 64, 128, 256\}$ one by one. For simplicity, only CoCNN2 was used to evaluate performance. Fig. 6 shows the average HR and NDCG values with changing dimension size for all data sets.

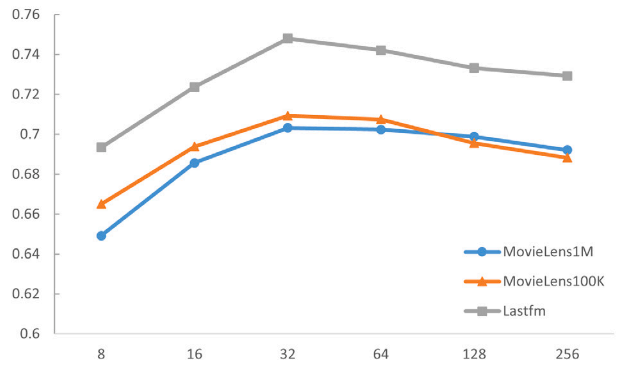
As shown in Fig. 6, we observe that on all data sets, both scores of our model exhibit a common trend in that their curves increase first and then decrease. In the beginning, embeddings with a larger dimensionality, containing more characters of items and preference information of users, improve model performance. Then, embeddings with too large dimensionality, too much redundant information increases the risk of over-fitting. Therefore, we conclude that an appropriate embedding size is indeed useful to improve the architecture.

4.2.4. Discover relationships

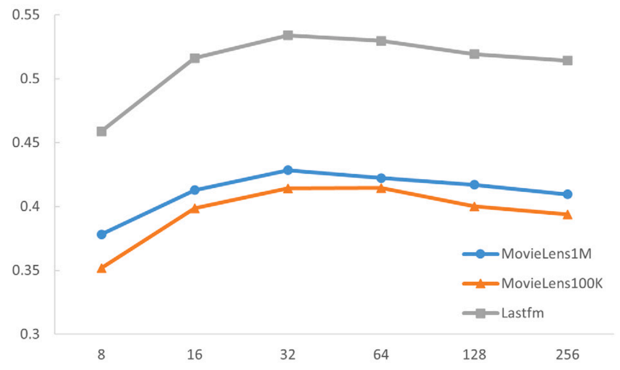
Finally, we further validated the relationship between the items in our model. In these experiments, we compared CoCNN with other personalized methods to show the ability to discover the relationship between items. For the trained embeddings of items, a Cosine function was used to measure the distance between the items. NeuMF has two

Table 4
Relationship between movies.

Movie name	Top	ONCF	BPR	CoFactor	CoCNN
Raiders of the Lost Ark	1	The Empire Strikes Back	The Empire Strikes Back	The Empire Strikes Back	The Empire Strikes Back
	2	Forrest Gump	Back to the Future	The Terminator	Braveheart
	3	Fallen	The Silence of the Lambs	Alien	Indiana Jones and the Last Crusade
	4	Alien	Forrest Gump	Young Frankenstein	Terminator 2: Judgment Day
The Godfather	1	Star Wars	Schindler's List	Dead Man Walking	Star Wars
	2	Return of the Jedi	Dead Man Walking	Star Wars	Fargo
	3	Fargo	Twelve Monkeys	Fargo	The Godfather: Part II
	4	Dead Man Walking	Star Wars	Return of the Jedi	Return of the Jedi
Star Trek	1	Star Trek V	Star Trek IV	Star Trek VI	Star Trek IV
	2	Star Trek V	Star Trek III	Star Trek III	Star Trek VI
	3	Incognito	Star Trek V	Star Trek V	Star Trek V
	4	Star Trek VI	Conan the Barbarian	Fallen	Star Trek III



(a) HR@10 scores v.s. dimension size



(b) NDCG@10 scores v.s. dimension size

Fig. 6. Effect of the dimension size on all data sets.

groups of embedding: MLP and GMF; thus, it is not suitable for calculating the similarity of items. For simplicity, only the MovieLens100K was used.

As seen from Table 4, with the MovieLens100K data sets, the top four items provided by CoCNN are more interpretable than the others. For example, given "Raiders of the Lost Ark" (an action and adventure style film), BPR, CoFactor, ONCF, and CoCNN offer users 3/4, 3/4, 2/4, and 4/4 movies of the same style, respectively. Similarly, given "The Godfather", our model offers users its sequel: "The Godfather: Part II." But other methods cannot obtain these results. "Star Trek" is one of the most popular series of movies. It has more sequels and a higher relative rated density, which provide methods easier access to learn similar features. In addition to other baselines, CoCNN achieves good results in finding similar movies. We conclude that the co-occurrence pattern is a sensible strategy for improving the performance of CF.

5. Conclusion

Most existing CF methods treat items independently and ignore the relationships among items. To solve this problem, we proposed a novel architecture, CoCNN, to provide personalized recommendations for users. In CoCNN, an embedding structure to capture the link between user–item and item–item was designed. Then, two ways of CNN structure were designed to directly apply to the embeddings. Finally, in recommendation tasks, extensive experiments were conducted to empirically evaluate the superior performance of CoCNN.

In the future, to improve our models, we will consider the following strategies: first, it is hard for neural networks to begin with meaningful features when they randomly initialized the weights. Thus, to get accurate latent features at the beginning of the training, a sound strategy is to incorporate some textual data, such as reviews and item descriptions, into our model. Also, for the sparse features of large-scale learning, it is important for neural models to learn feature interactions. Thus, to improve our model, we plan to combine features in low-dimensional dense embeddings.

CRediT authorship contribution statement

Ming Chen: Investigation, Data curation, Writing – original draft, Project administration. **Tianyi Ma:** Data curation, and Software. **Xiuze Zhou:** Conceptualization of this study, Methodology, Validation, Investigation, Data curation, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The authors would like to thank Michael McAllister for proofreading this manuscript.

References

Al-Ayyoub, M., Nuseir, A., Alsmearat, K., Jararweh, Y., & Gupta, B. (2018). Deep learning for Arabic NLP: A survey. *Journal of Computational Science*, *26*, 522–531.
Benhamdi, S., Babouri, A., & Chiky, R. (2017). Personalized recommender system for e-learning environment. *Education and Information Technologies*, *22*(4), 1455–1477.
Cai, Q., Filos-Ratsikas, A., Tang, P., & Zhang, Y. (2018). Reinforcement mechanism design for e-commerce. In *Proceedings of the 27th international conference on World Wide Web* (pp. 1339–1348).
Chen, M., Li, Y., & Zhou, X. (2021). CoNet: Co-occurrence neural networks for recommendation. *Future Generation Computer Systems*, *124*, 308–314.
Chen, H., Xin, X., Wang, D., & Ding, Y. (2021). Decomposed collaborative filtering: modeling explicit and implicit factors for recommender systems. In *Proceedings of the 14th ACM international conference on web search and data mining* (pp. 958–966).

Chen, M., & Zhou, X. (2020). DeepRank: Learning to rank with neural networks for recommendation. *Knowledge-Based Systems*, *209*, Article 106478.

Cheng, H. -T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., et al. (2016). Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems* (pp. 7–10).

Choi, M., Jeong, Y., Lee, J., & Lee, J. (2021). Local collaborative autoencoders. In *Proceedings of the 14th ACM international conference on web search and data mining* (pp. 734–742).

Costa, F. S. d., & Dolog, P. (2019). Collective embedding for neural context-aware recommender systems. In *Proceedings of the 13th ACM conference on recommender systems* (pp. 201–209).

Covington, P., Adams, J., & Sargin, E. (2016). Deep neural networks for Youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems* (pp. 191–198).

Du, Z., Tang, J., & Ding, Y. (2019). POLAR++: Active one-shot personalized article recommendation. *IEEE Transactions on Knowledge and Data Engineering*.

Ebesu, T., Shen, B., & Fang, Y. (2018). Collaborative memory network for recommendation systems. In *Proceedings of the 41st international ACM SIGIR conference on research & development in information retrieval* (pp. 515–524).

Harper, F. M., & Konstan, J. A. (2015). The Movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, *5*(4), 1–19.

He, X., Du, X., Wang, X., Tian, F., Tang, J., & Chua, T. (2018). Outer product-based neural collaborative filtering. Proceedings of the 27th international joint conference on artificial intelligence. (pp. 2227–2233).

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. (2017). Neural collaborative filtering. In *Proceedings of the 26th international conference on World Wide Web* (pp. 173–182).

Hongtao, L., & Qinchuan, Z. (2016). Applications of deep convolutional neural network in computer vision. *Journal of Data Acquisition and Processing*, *31*(1), 1–17.

Hu, L., Li, C., Shi, C., Yang, C., & Shao, C. (2020). Graph neural news recommendation with long-term and short-term interest modeling. *Information Processing & Management*, *57*(2), Article 102142.

Khan, S., Rahmani, H., Shah, S. A. A., & Bennamoun, M. (2018). A guide to convolutional neural networks for computer vision. *Synthesis Lectures on Computer Vision*, *8*(1), 1–207.

Kim, D., Park, C., Oh, J., Lee, S., & Yu, H. (2016). Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM conference on recommender systems* (pp. 233–240).

Koren, Y., Bell, R. M., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, *42*(8), 30–37.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Li, D., Chen, C., Lu, T., Chu, S., & Gu, N. (2019). Mixture matrix approximation for collaborative filtering. *IEEE Transactions on Knowledge and Data Engineering*.

Liang, D., Altosaar, J., Charlin, L., & Blei, D. M. (2016). Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the 10th ACM conference on recommender systems* (pp. 59–66).

Lin, Y., Feng, S., Zeng, W., Lin, F., Liu, Y., & Wu, P. (2021). Adaptive course recommendation in MOOCs. *Knowledge-Based Systems*, Article 107085.

Liu, S., Xu, S., Yu, W., Fu, Z., Zhang, Y., & Marian, A. (2021). FedCT: Federated collaborative transfer for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval* (pp. 716–725).

Lu, S., Chen, H., Zhou, X., Wang, B., Wang, H., & Hong, Q. (2018). Graph-based collaborative filtering with MLP. *Mathematical Problems in Engineering*, *2018*.

Mou, L., Meng, Z., Yan, R., Li, G., Xu, Y., Zhang, L., et al. (2016). How transferable are neural networks in NLP applications? In *Proceedings of the 2016 conference on empirical methods in natural language processing* (pp. 479–489).

Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidtthieme, L. (2012). BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th conference on uncertainty in artificial intelligence* (pp. 452–461).

Rendle, S., Krichene, W., Zhang, L., & Anderson, J. (2020). Neural collaborative filtering vs. Matrix factorization revisited. arXiv preprint arXiv:2005.09683.

Sarwar, B. M., Karypis, G., Konstan, J. A., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web* (pp. 285–295).

Schedl, M., Knees, P., McFee, B., Bogdanov, D., & Kaminskas, M. (2015). Music recommender systems. In *Recommender systems handbook* (pp. 453–492). Springer.

Sedhain, S., Menon, A. K., Sanner, S., & Xie, L. (2015). Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web* (pp. 111–112).

Shen, D., Wu, G., & Suk, H. -I. (2017). Deep learning in medical image analysis. *Annual Review of Biomedical Engineering*, *19*, 221–248.

Shen, X., Yi, B., Liu, H., Zhang, W., Zhang, Z., Liu, S., et al. (2019). Deep variational matrix factorization with knowledge embedding for recommendation system. *IEEE Transactions on Knowledge and Data Engineering*.

Song, S., Yu, F., Zeng, A., Chang, A. X., Savva, M., & Funkhouser, T. (2017). Semantic scene completion from a single depth image. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1746–1754).

Strub, F., & Mary, J. (2015). Collaborative filtering with stacked denoising autoencoders and sparse inputs. In *Proceedings of the 29th NIPS workshop on machine learning for ecommerce*.

Sun, Z., Yang, J., Zhang, J., Bozzon, A., Chen, Y., & Xu, C. (2017). MRLR: Multi-level representation learning for personalized ranking in recommendation. In *Proceedings of the 26th international joint conference on artificial intelligence* (pp. 2807–2813).

Van Den Oord, A., Dieleman, S., & Schrauwen, B. (2013). Deep content-based music recommendation. In *Neural information processing systems conference*: *Vol. 26*, Neural Information Processing Systems Foundation (NIPS).

Wang, J., Gao, N., Peng, J., & Mo, J. (2019). DCAR: Deep collaborative autoencoder for recommendation with implicit feedback. In *International conference on artificial neural networks* (pp. 172–184). Springer.

Wang, S., Huang, S., Liu, T. -Y., Ma, J., Chen, Z., & Veijalainen, J. (2016). Ranking-oriented collaborative filtering: A listwise approach. *ACM Transactions on Information Systems*, *35*(2), 1–28.

Wang, X., Wang, R., Shi, C., Song, G., & Li, Q. (2020). Multi-component graph convolutional collaborative ciltering. In *Proceedings of the 34th aaai conference on artificial intelligence*: *Vol. 34*, (pp. 6267–6274).

Wang, R., Wu, Z., Lou, J., & Jiang, Y. (2022). Attention-based dynamic user modeling and deep collaborative filtering recommendation. *Expert Systems with Applications*, *188*, Article 116036.

Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., & Tan, T. (2019). Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*: *Vol. 33*, (pp. 346–353).

Wu, C., Wu, F., An, M., Huang, J., Huang, Y., & Xie, X. (2019). Neural news recommendation with attentive multi-view learning. In *Proceedings of the 28th international joint conference on artificial intelligence* (pp. 3863–3869). International Joint Conferences on Artificial Intelligence Organization.

Wu, X., Zeng, W., Lin, F., & Zhou, X. (2021). NeuRank: Learning to rank with neural networks for drug–target interaction prediction. *BMC Bioinformatics*, *22*(1), 1–17.

Wu, H., Zhou, Q., Nie, R., & Cao, J. (2020). Effective metric learning with Co-occurrence embedding for collaborative recommendations. *Neural Networks*, *124*, 308–318.

Xue, H., Dai, X., Zhang, J., Huang, S., & Chen, J. (2017). Deep matrix factorization models for recommender systems. In *Proceedings of the 26th international joint conference on artificial intelligence* (pp. 3203–3209).

Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., & Sang, N. (2018). Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision* (pp. 325–341).

Zhang, L., Liu, P., & Gulla, J. A. (2019). Dynamic attention-integrated neural network for session-based news recommendation. *Machine Learning*, *108*(10), 1851–1875.

Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys*, *52*(1), 1–38.

Zhao, X., Xia, L., Zou, L., Liu, H., Yin, D., & Tang, J. (2020). Whole-chain recommendations. In *Proceedings of the 29th ACM international conference on information & knowledge management* (pp. 1883–1891).

Zhou, Y., Huang, C., Hu, Q., Zhu, J., & Tang, Y. (2018). Personalized learning full-path recommendation model based on LSTM neural networks. *Information Sciences*, *444*, 135–152.

Zhou, X., & Wu, S. (2016). Rating LDA model for collaborative filtering. *Knowledge-Based Systems*, *110*(01), 135–143.

Zhu, Y., Li, H., Liao, Y., Wang, B., Guan, Z., Liu, H., et al. (2017). What to do next: Modeling user behaviors by time-LSTM. In *Proceedings of the 26th international joint conference on artificial intelligence*: *Vol. 17*, (pp. 3602–3608).

Zhu, Y., Wu, X., Qiang, J., Yuan, Y., & Li, Y. (2021). Representation learning with collaborative autoencoder for personalized recommendation. *Expert Systems with Applications*, *186*, Article 115825.