Contents lists available at ScienceDirect

Future Generation Computer Systems

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

CoNet: Co-occurrence neural networks for recommendation

Ming Chen [a], Yunhao Li [b], Xiuze Zhou [a,*]

[a] Hithink RoyalFlush Information Network Co., Ltd., Hangzhou 310023, China
[b] Faculty of Computing, Harbin Institute of Technology, Harbin 150006, China

ARTICLE INFO

Article history:
Received 24 October 2020
Received in revised form 16 May 2021
Accepted 7 June 2021
Available online 9 June 2021

Keywords:
Recommender systems
Neural network
Collaborative Filtering
Co-occurrence pattern
Attention network

ABSTRACT

Assuming that both users and items are independent and identically distributed, most existing methods model user–item pairs, while ignoring the relationship between items, leading to limited performance. To solve this problem, we propose a novel neural network, CoNet, which can effectively model the co-occurrence pattern for Collaborative Filtering (CF). We argue that items always occur in pairs, i.e. an item co-occurrence pattern. For example, movies "Harry Potter 1" and "Harry Potter 2" are always viewed by users who like magic style films. To learn the latent features, CoNet is simultaneously modeled on user–item and item–item interactions. Compared with methods that train on a single user–item pair, CoNet can encode highly descriptive features from the co-occurrence pattern.

To achieve a better performance, we design an attention network to learn the weight of a user's preference for different items and subsequently aggregate the weighted embeddings to obtain the co-occurrence representations. Finally, we conducted extensive experiments using several data sets, which show that the proposed method is superior to other baseline approaches. Source code of CoNet is available from https://github.com/XiuzeZhou/conet.

1. Introduction

Recommender systems, integrated into many applications such as e-commerce [1,2], entertainment [3,4] and social networks [5,6], provide personalized recommendation services for users. To alleviate the information overloading problem, recommender systems help users find the best-suited service or product among a plethora of options [7,8].

Matrix Factorization (MF) [9] learns low-dimensional latent features to represent each user and item [10]. Most MF-based Collaborative Filtering (CF) methods combine user and item latent features linearly, which results in limited performance when dealing with highly complicated real-world data [11,12]. For better performance, many deep learning architectures have been proposed to capture nonlinear relationships encoded by their hidden layers. For example, Convolutional Neural Network (CNN), which has achieved outstanding achievements in Computer Vision (CV), is also applied to learn high-order correlations among latent features for recommendation [13,14].

To extract the meaningful features from the raw data, effective feature extraction methods, such as Variational Auto-Encoder (VAE) [15,16] and Denoising Auto-Encoder (DAE) [17–19], are adopted. Multi-Layer Perceptron (MLP), the most commonly used neural network architecture, learns deep and non-linear representations [11,20]. Through hidden layers of neural networks, those methods effectively capture high-order information from user–item interactions. Those methods, widely applied to recommender systems, have achieved great success.

However, most CF-based methods assume that all items are independently, and separate all examples into distinct, unrelated instances. Those methods learn representation from user–item interactions. For one time, only a single user–item pair is used for training. However, the methods face one major challenge, i.e. the coupling relationships between items is ignored during training.

Inspired by Liang et al. [21], we designed a novel neural network architecture, CoNet, which takes the advantage of item co-occurrence from user–item interaction data. We assume that items usually occur in pairs, i.e. an item co-occurrence pattern, which is a simple but effective way to enhance features learning.

Fig. 1 illustrates a toy-example of movie co-occurrence patterns. When one user (who likes magic style films) always watches "Harry Potter 1" and "Harry Potter 2", it is highly probably that user will give the same rating to both. Our model learns features not only from user–item interactions (as is the case with most methods), but also from item–item co-occurrence information, a powerful ability to capture the relationship between items.

Another important assumption for co-occurrence modeling is that the more frequently the two items occur together, the more similar they are. For example, in users' records, movies "Harry

* Corresponding author.
E-mail addresses: chm@zju.edu.cn (M. Chen), crimsonjhu@gmail.com (Y. Li), zhouxiuze@foxmail.com (X. Zhou).

https://doi.org/10.1016/j.future.2021.06.008
0167-739X/© 2021 Elsevier B.V. All rights reserved.

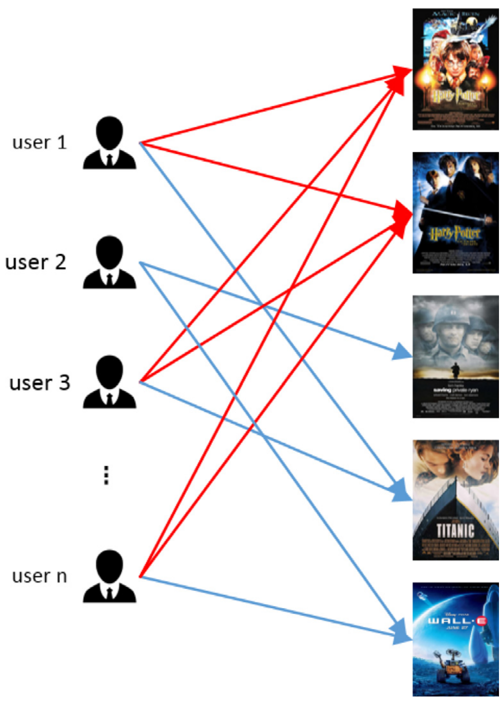


Fig. 1. Example of movie co-occurrence patterns.

Potter 1" and "Harry Potter 2" always appear together. Thus, we argue that both movies are similar and close to each other. By training on co-occurrence patterns, the relationship between items can be better discovered.

Also, in recommendation tasks, to learn latent features, many existing neural network-based approaches [11,13,22] model complex structures of user–item interactions. However, the performance of neural network-based approaches is limited by the assumption that all items of a user contribute equally during the interaction modeling process. To value users' preferences for items, in deep leaning models, an attention network is widely applied. For example, Cao et al. [23] applied an attention network to learn a aggregation strategy for group recommendation tasks; Chen et al. [24] used an attention-based network to give explanations to users; Xiao et al. [12] proposed attention networks in factorization machines to learn the weights of features. Therefore, the attention mechanism is a promising technique for valuing users' preferences for their items.

Thus, to capture users' interests in different items, we incorporated an attention network to our model to weigh users' different preferences. When one item is compared with another, users have distinct preferences towards one, although both items are rated equally. The attention network places higher values on the item that might have received a higher interest rating from the user, thereby providing a new perspective for exhibiting a close connection between the two items.

We show that there are three advantages to the proposed model:

(1) It performs well when a co-occurrence pattern is used to train our model. To learn the latent features effectively, our model is simultaneously modeled on user–item and item–item interactions. Compared with methods that train on a single user–item pair, our model learns more useful features from the co-occurrence pattern;

(2) Rather than treating preferences uniformly, the attention mechanism is used to automatically assigns weights to value a user's different preferences over items;

(3) Because it explicitly captures the latent relationship between items, it is comparatively descriptive. Compared with the existing methods, our model provides more interpretable results.

2. Preliminaries

Given a user set $U = \{u_1, u_2, \ldots, u_n\}$ and an item set $I = \{i_1, i_2, \ldots, i_m\}$, the interaction matrix $\boldsymbol{R} \in \mathbb{R}^{n\times m}$ is defined:

$$r_{ui} = \begin{cases} 1, & \text{if a observed interaction between } u \text{ and } i; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Then, generated from $\boldsymbol{R}$, we construct a new interaction matrix, $\boldsymbol{Y} \in \mathbb{R}^{n\times m\times m}$, about users and their item pairs, as follows:

$$y_{uij} = \begin{cases} 1, & \text{if } \ r_{ui} = 1 \text{ and } r_{uj} = 1 \\ 0, & \text{otherwise} \end{cases}, \tag{2}$$

where $y_{uij} = 1$ denotes that user, u, interacts with items i and j, i.e. items i and j are co-occurring for user u; $y_{uij} = 0$ denotes that in user u's historical record, no interaction is observed between items i and j.

3. Related work

First, in addition to some background information, we review MF, one of the most widely used methods for CF. Then, we briefly introduce a neural network method, Neural Collaborative Filtering (NCF).

3.1. Matrix factorization (MF)

Let $\boldsymbol{P} \in \mathbb{R}^{k\times n}$ and $\boldsymbol{Q} \in \mathbb{R}^{k\times m}$ denote the low-rank latent features of users and items, respectively, where k denotes the number of latent features. $\boldsymbol{p}_u \in \mathbb{R}^{k\times 1}$ and $\boldsymbol{q}_i \in \mathbb{R}^{k\times 1}$ denote the latent features of user, u, and item, i, respectively. Given a set of observed instances, $\boldsymbol{O}$, the goal of MF is to learn $\boldsymbol{P}$ and $\boldsymbol{Q}$ to reconstruct $\boldsymbol{R}$:

$$\mathcal{L}_1 = \sum_{(u,i)\in \boldsymbol{O}} \left(r_{u,i} - \boldsymbol{p}_u^T \boldsymbol{q}_i\right)^2 + \lambda \left(\|\boldsymbol{P}\|_F^2 + \|\boldsymbol{Q}\|_F^2\right), \tag{3}$$

where λ is a regularization parameter; and $\|\cdot\|_F^2$ denotes the Fibonacci-norm.

3.2. Neural collaborative filtering (NCF)

One problem with MF-based methods is that they cannot capture high-order features from user–item interactions [11,25]. Neural network is a good method for learning complex and high-order features from data [26]. NCF, a MLP-based framework for CF, is shown in Fig. 2.

A user–item pair (u, i) as a sparse input, is mapped to embedding vectors to obtain their dense representation and then fed to MLP to learn deep structures on user–item interactions. Finally, the user–item pair obtains its prediction, $\hat{r}_{ui}$, from the last hidden layer of MLP, formulated as follows:

$$\hat{r}_{ui} = f_{MLP}\left(\boldsymbol{p}_u, \boldsymbol{q}_i\right), \tag{4}$$

Then, to measure the loss of all user–item pairs, we use *binary cross-entropy loss*:

$$\mathcal{L} = -\sum_{(u,i)\in \boldsymbol{O}\cup\boldsymbol{O}^-} r_{ui} log\widehat{r}_{ui} + (1 - r_{ui})\, log(1 - \widehat{r}_{ui}), \tag{5}$$

where $\boldsymbol{O}^-$ denotes the set of negative instances.

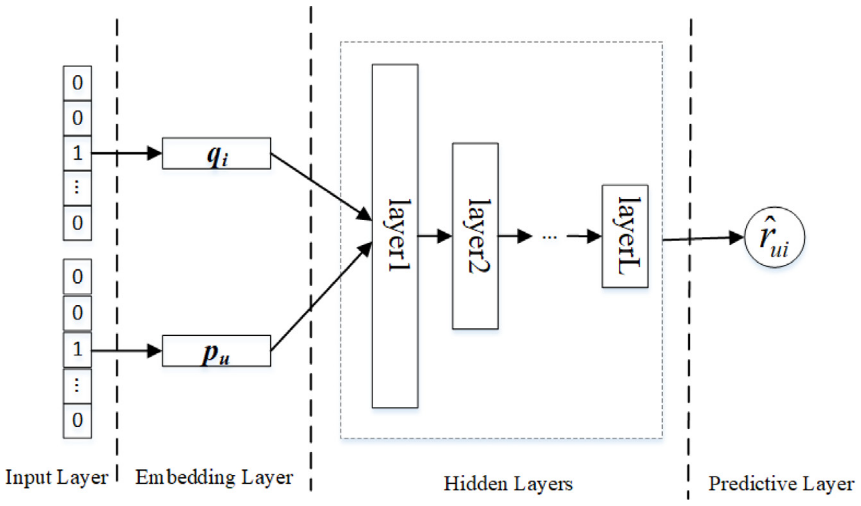


Fig. 2. Framework of NCF.

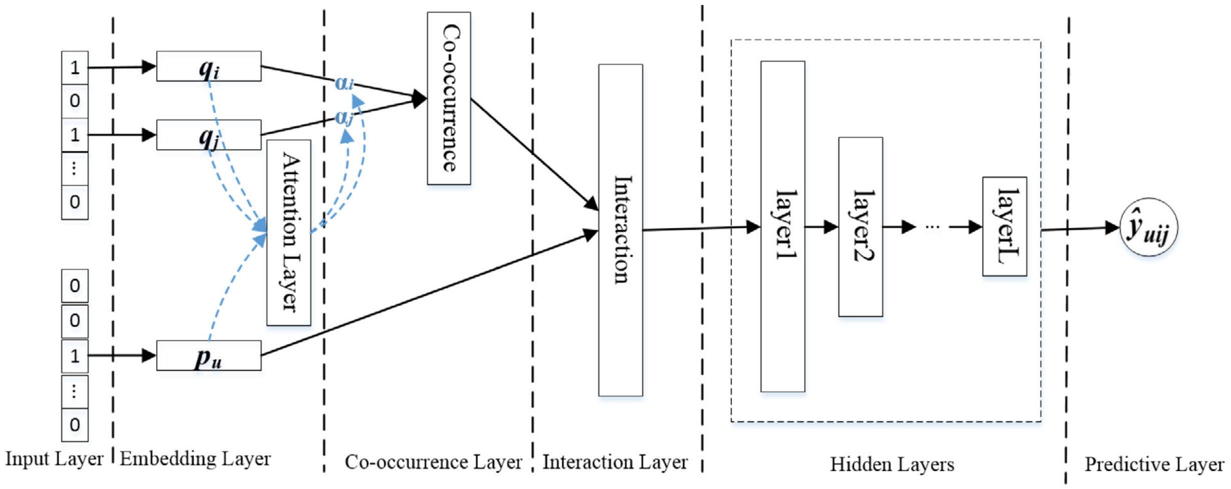


Fig. 3. Framework of CoNet.

4. The proposed method

We design a neural network: CoNet, for implicit feedback. From the perspective of users, items are co-occurring, and using items co-occurrence pattern for CoNet, called items CoNet. This is our primary perspective; therefore, in the rest of this paper, we treat CoNet as an items CoNet. Similarly, from the perspective of items, users are co-occurring, CoNet also can be modeled on a users co-occurrence pattern.

4.1. Architecture

CoNet learns its latent features from user–item and item–item interactions. CoNet aims at predicting the probability that a user will be interested in an item after CoNet is trained on a co-occurrence pattern. Fig. 3 illustrates the neural network framework of CoNet, which consists of the following seven layers: input, embedding, attention, co-occurrence, interaction, hidden, and predictive.

Input and Embedding Layers. The embedding layer of a neural network aims to convert an input from a sparse representation into a dense one, defined as follows:

$$\boldsymbol{p}_u = f_{lookup}(u), \tag{6}$$

$$\boldsymbol{q}_i = f_{lookup}(i), \tag{7}$$

Co-occurrence Layer. The co-occurrence layer aims to obtain co-occurrence vectors by compressing two co-occurrence embeddings into one. Co-occurrence pattern, a key factor to better formulate the personalized ranking problem helps model to learn mutual correlations between cross users/items. The item–item co-occurrence vector is computed by aggregating two item embeddings, defined as follows:

$$\boldsymbol{v}_{ij} = f_c\left(\boldsymbol{q}_i, \boldsymbol{q}_j\right), \tag{8}$$

$$s.t.\ \boldsymbol{q}_i = f_c\left(\boldsymbol{q}_i, \boldsymbol{q}_i\right), \tag{9}$$

where $\boldsymbol{v}_{ij}$ denotes the co-occurrence feature of items i and j; $f_c(\cdot)$ denotes the function to calculate the co-occurrence features from embeddings $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$. Without attention network, *arithmetic mean* is a simple and effective method to obtain $\boldsymbol{v}_{ij}$:

$$\boldsymbol{v}_{ij} = \frac{\boldsymbol{q}_i + \boldsymbol{q}_j}{2}. \tag{10}$$

Attention Layer. The attention layer aims to learn different weights of items that interact with users and then aggregate weighted embeddings to generate their co-occurrence representation. The attention network learns the importance of each of the items and assigns proper weights to them, representing user preferences over items.

Rather than applying a predefined strategy, we adopt a new aggregation strategy to obtain the interaction between a user and his items. To learn the weight of different items that interact with users, an attention network is designed.

We assume that a user, when comparing one item with another, tends to pay different interest in one of the items, although the user rates each of the items the same when rating them separately. Therefore, in CoNet, rather than treating the items uniformly and separately, an attention mechanism is used to learn the weights of each pair-wise interaction.

CoNet calculates a weighted sum for a pair of item representations, where weights are adaptively learned from the attention neural. It places a higher value on the preferred item, defined as follows:

$$\boldsymbol{v}_{ij} = \alpha_i \boldsymbol{q}_i + (1 - \alpha_i)\boldsymbol{q}_j, \tag{11}$$

where $\boldsymbol{v}_{ij}$ denotes the co-occurrence feature from weighted embeddings; α_i denotes the relative preference degree for item, i, of user, u, when facing items i and j. The attention score for user u and item i is defined as follows:

$$s_{ui} = f_a\left(\boldsymbol{p}_u, \boldsymbol{q}_i\right), \tag{12}$$

where $f_a(\cdot)$ denotes the function of the attention network. We use $\boldsymbol{p}_u^T \boldsymbol{q}_i$ as the attention score for user u and item i. Then, the weight of the contribution of the attention network for co-occurrence vectors is defined as follows:

$$\begin{aligned} \alpha_i &= \frac{exp\left(s_{ui}\right)}{exp\left(s_{ui}\right) + exp\left(s_{uj}\right)} \\ &= \frac{1}{1 + exp\left(s_{uj} - s_{ui}\right)} \\ &= \sigma\left(s_{ui} - s_{uj}\right), \end{aligned} \tag{13}$$

where $\sigma(\cdot)$ is the *sigmoid* function.

Interaction Layer. The interaction layer aims to capture the row-rank relationships by modeling user–item interactions in the shallow layer:

$$\boldsymbol{h}_0 = f_i\left(\boldsymbol{p}_u, \boldsymbol{q}_i\right), \tag{14}$$

where $f_i(\cdot)$ denotes the interaction functions between $\boldsymbol{p}_u$ and $\boldsymbol{q}_i$, such as *concatenation*, *element-wise product*, and *element-wise sum*. We chose *concatenation* as our interaction function.

Hidden Layers. Hidden layers aim to capture high-rank relationships and nonlinear correlations between users and items. Hidden layers provide neural networks a powerful ability to model the high-rank relationships between features as follows:

$$\boldsymbol{h}_1 = a\left(\boldsymbol{W}_1^T \boldsymbol{h}_0 + \boldsymbol{b}_1\right), \tag{15}$$

$$\cdots$$

$$\boldsymbol{h}_L = a\left(\boldsymbol{W}_L^T \boldsymbol{h}_{L-1} + \boldsymbol{b}_L\right), \tag{16}$$

where $\boldsymbol{W}_l$, $\boldsymbol{b}_l$, and $\boldsymbol{h}_l$ denote weight, bias, and output of the lth $(0 < l \le L)$ layer, respectively; $a(\cdot)$ denotes activation function.

Predictive Layer. This layer aims to map the outcome of the final hidden layer to predict probability, $\hat{y}_{uij}$, formulated as follows:

$$\hat{y}_{uij} = \sigma\left(\boldsymbol{W}_p^T \boldsymbol{h}_L + \boldsymbol{b}_p\right), \tag{17}$$

where $\boldsymbol{W}_p$ and $\boldsymbol{b}_p$ denote weight and bias, respectively; $\sigma(\cdot)$ denotes the activation function, *sigmoid*.

After the network is well trained, we predict the possibility, $\hat{y}_{ui}$, of user u on item i. *Arithmetic mean*, Eq. (10), is used to replace the attention mechanism, i.e. set $\boldsymbol{q}_i = \boldsymbol{q}_j$ to obtain $\boldsymbol{v}_{ij} = \boldsymbol{q}_i$. Then, (u, i) is fed to our neural network to make predictions.

Table 1
Statistics for all data sets.

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

Learning. *Cross-entropy* is used to evaluate loss and the L2 norm is used to regularize all learning parameters. Then, we define the objective function:

$$\mathcal{L} = -\sum_{u=1}^{n} \sum_{(i,j)\in \boldsymbol{C}\cup\boldsymbol{C}^-} y_{uij} log\widehat{y}_{uij} + \left(1 - y_{uij}\right) log\left(1 - \widehat{y}_{uij}\right), \tag{18}$$

where $\boldsymbol{C}$ and $\boldsymbol{C}^-$ denote the set of observed and unobserved interactions in $\boldsymbol{Y}$, respectively.

Similarly, from the perspective of items, CoNet modeled on a user co-occurrence pattern is designed in a same framework.

5. Experiments

First, we describe the experimental settings, including data sets and preprocessing, evaluation metrics, baseline approaches, and parameter settings. Then, we experiment with the following questions one-by-one:

-RQ1: Compared with other methods with a single user–item pair, how does our model perform?

-RQ2: How effective is our designed attention network? Does our network provide better performance with it than without it?

-RQ3: Can our model discover the relationship between items?

5.1. Experimental setting

Data Sets and Preprocessing. Experiments were conducted using three publicly available data sets: Lastfm, MovieLens100K, and MovieLens1M. Lastfm collected from Last.fm, is available from the GroupLens web site.[1] MovieLens100K and MovieLens1M collected by GroupLens [27], are available from its web site.[2] Some statistics are shown in Table 1.

Baseline Approaches. Conet is compared against the following baselines:

- **ItemPop** [28], a non-personalized method model, uses the number of interactions of items to rank;
- **BPR** [29], one popular MF-based methods for ranking learning, provides personalized ranking by optimizing a pair-wise loss function;
- **NCF** [11], a neural network architecture, MLP, is used to make predictions;
- **NeuMF** [11], a neural network method, which combines a linear kernel, learns low-rank features by Generalized Matrix Factorization (GMF) and learns high-rank features by NCF from user–item interactions;
- **DeepCF** [30], a neural network ranking learning model, in which the row, $\boldsymbol{R}_{\boldsymbol{u}*}$, is used to represent user u, and the column, $\boldsymbol{R}_{*\boldsymbol{i}}$, is used to represent item i.

[1] https://grouplens.org/datasets/hetrec-2011/.

[2] https://grouplens.org/datasets/movielens/.

Table 2
NDCG@10 and *HR@10* scores of all models.

		ItemPop	NCF	BPR	NeuMF	DeepCF	CoNet
MovieLens100K	HR@10	0.4163	0.6628	0.6801	0.6886	0.6932	0.7063
	NDCG@10	0.2407	0.3914	0.3949	0.4008	0.4093	0.4182
MovieLens1M	HR@10	0.4638	0.6704	0.6932	0.7045	0.7014	0.7083
	NDCG@10	0.2694	0.4077	0.4123	0.4201	0.4222	0.4304
Lastfm	HR@10	0.4822	0.7089	0.7044	0.7111	0.7251	0.7480
	NDCG@10	0.3378	0.4926	0.4923	0.5096	0.5185	0.5341

Parameter Settings. Comparing the models with a fixed embedding dimension is more meaningful [31]. Therefore, we set some key parameters in all models to the same value: learning rate, regularization coefficient (λ), and embedding size (latent feature size) are set at 0.001, 10^{-5}, and 64, respectively; for all neural network models, batch size and epochs are set at 1024 and 32, respectively.

Adam [32], was chosen the optimizer for our objective function. We reported the performance of all results without pretraining and determined their average values after randomly running five times using the same settings. All personalized methods were implemented by TensorFlow. To ensure the generalization, 8 to 64 negative examples were sampled to pair with one positive example.

Evaluation Metrics. To evaluate ranking performance, the *leave-one-out* evaluation was used by Deng et al. [30], Song et al. [22], He et al. [11], Xue et al. [33], and Rendle et al. [29]. For each user, the latest rated item and 100 randomly sampled unrated ones were selected as testing data. Then, we adopted *NDCG* and Hit Ratio (*HR*) to measure the top-n performance. *NDCG@n* is defined as follows:

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

$$DCG@n = \sum_{i=1}^{n} \frac{2^{r_i} - 1}{log(1+i)}$$

$$r_i = \begin{cases} 1, & \text{if the item at position } i \text{ is a hit item} \\ 0, & \text{other} \end{cases}$$

where *IDCG@n* denotes the maximum possible value of *DCG*.

Then, given a list with *n* items, we define the *HR@n* as follows:

$$HR@n = \frac{hits}{n}$$

where *hits* is the number of times items in the list for each user.

5.2. Overall performance (RQ1)

First, we conducted experiments using three data sets, and the performance of all methods is shown in Table 2. From Table 2, we observe the following: (1) In terms of both *HR* and *NDCG*, the results of all methods are consistent on different data sets; (2) Compared with the results of other baseline approaches using the same data set, CoNet always achieves the best results and performs considerably higher NDCG and HR values than other baseline methods. The sequence of average performance (ItemPop $<$ BPR $<$ NCF $<$ NeuMF $<$ DeepCF $<$ CoNet) suggests that our model can make good predictions with co-occurrence patterns; (3) For *HR@10* and *NDCG@10*, a comparison of the results for NCF and NeuMF (models built on user–item pairs) and CoNet shows that CoNet achieves significantly best performance on MovieLens100K, MovieLens1M, and Lastfm. The best results indicate that the co-occurrence pattern is effective for models to learn more meaningful embeddings; (4) Similarly, a comparison of the results for DeepCF (a model built on user and item vectors) and CoNet shows that too many users and items involved in modeling do not guarantee good effect. For too many factors improve the effect of DeepCF at the same time bring a lot of noise, which degrades its performance. In summary, for tackling CF tasks, CoNet trained with co-occurrence patterns performs well.

5.3. Effect of attention network (RQ2)

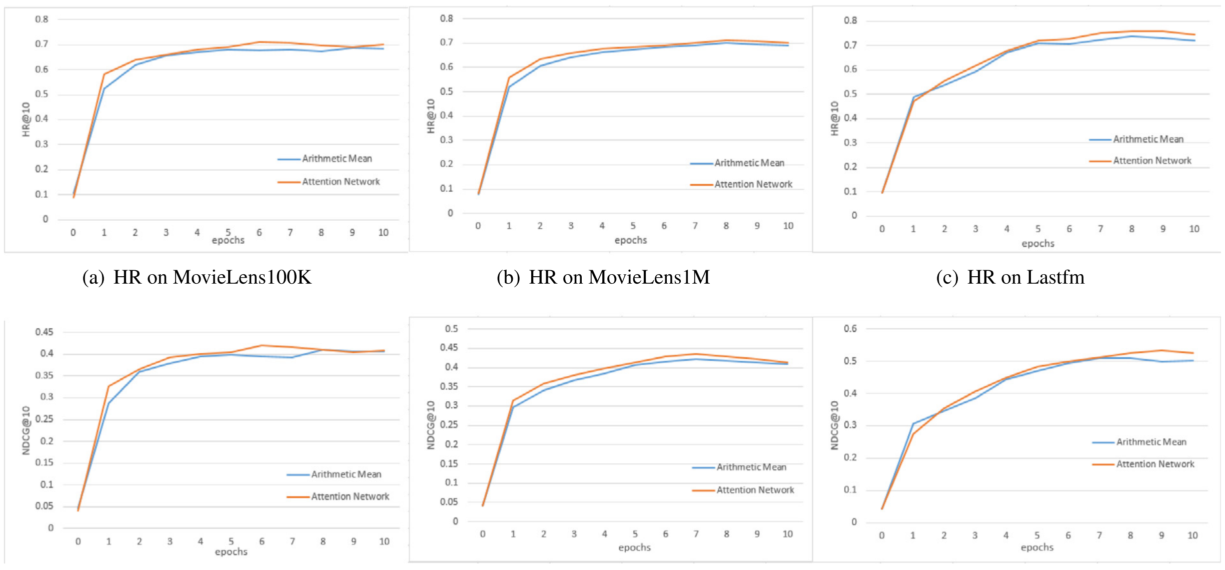
To fully explore the capacity of our model, we investigate the effect of an attention network for recommendation. An attention network provides a strong force for users to evaluate their preference levels. The weights learned from the attention network are used to estimate the co-occurrence features. The goal of the attention network is to learn each user's pair-wise preferences for different items.

In these experiments, for comparison, we used *arithmetic mean* instead of the attention network, and the performance of CoNet with and without the attention network (See Fig. 4). As seen from Fig. 4, in general, on all data sets, the results for *HR* and *NDCG* are consistent. As the number epoch increases, all the methods improve in predicting ranking performance. Also, CoNet with the attention network, always achieves higher *NDCG* and *HR* scores than itself with *arithmetic mean* under the same settings, showing the good performance of the attention network for our model. Therefore, we conclude that adding the attention mechanism, which is critical for good predicting, enables our model to provide higher quality recommendations.

5.4. Discover the relationship (RQ3)

Finally, we further validated the relationship between the items in our model. In these experiments, we compared CoNet with other personalized methods to show their ability to discover the relationship between items. A Cosine function was used to measure the distance between the items of the trained embeddings of items. NeuMF has two groups of embedding: MLP and GMF, so it is not suitable for calculating the similarity of items. For simplicity, only the MovieLens100K was used.

As seen from Table 3, with the MovieLens100K data sets, the top-4 items provided by CoNet are more interpretable than the others. For example, given Raiders of the Lost Ark, an action and adventure style film, BPR, NCF, DeepCF, and CoNet offer users 3/4, 1/4, 3/4, and 4/4 movies of the same style, respectively. Also, CoNet recommends users the sequel to the film: Indiana Jones and the Last Crusade. Similarly, given The Godfather, our model offers its sequel: The Godfather: Part II, to users. But other methods cannot obtain these results. Star Trek is one of the most popular series of movies. It has more sequels, a higher relative rated density, which helps methods to easier access to learn similar features. In addition to NCF, the other three models can achieve good results that all of them can find similar movies. We conclude that the co-occurrence pattern is a sensible strategy for improving the performance of CF.



(a) HR on MovieLens100K (b) HR on MovieLens1M (c) HR on Lastfm

(d) NDCG on MovieLens100K (e) NDCG on MovieLens1M (f) NDCG on Lastfm

Fig. 4. Effect of attention network.

Table 3
Relationship between movies.

Movie name	top	NCF	BPR	DeepCF	CoNet
Raiders of the Lost Ark	1	The Empire Strikes Back	The Empire Strikes Back	The Empire Strikes Back	The Empire Strikes Back
	2	The Terminator	Back to the Future	Alien	Braveheart
	3	Alien	The Silence of the Lambs	Fallen	Indiana Jones and the Last Crusade
	4	Blade Runner	Forrest Gump	Young Frankenstein	Terminator 2: Judgment Day
The Godfather	1	Fargo	Schindler's List	Dead Man Walking	Star Wars
	2	Star Wars	Dead Man Walking	Star Wars	Fargo
	3	Return of the Jedi	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 IV	Star Trek IV	Star Trek VI	Star Trek III
	2	Spawn	Star Trek III	Star Trek III	Star Trek VI
	3	Star Trek VI	Star Trek V	Star Trek V	Star Trek V
	4	Incognito	Conan the Barbarian	Fallen	Judge Dredd

6. Conclusion

We proposed a novel neural network framework, CoNet, with a powerful ability to represent and cover user preferences from a co-occurrence pattern, which contains more informative interactions between users and items. Unlike existing CF methods modeled on a single user–item pair, to make predictions, we modeled on user–item and item–item interactions simultaneously. To learn a personalized weight preference for each user, an attention network is added to CoNet. The attention network provides an adaptive weighting strategy for CoNet to learn a user's preference level for items. The experimental results show that CoNet achieves quite outstanding performance for high quality recommendations.

In our future work, first, to enrich latent features, we will explore incorporating auxiliary information, such as textual and visual information, into our model. Second, interaction functions are important for our models to learn the relationships between user–item and item–item. Thus, a more effective function is need to be studied further. Finally, to further improve recommendation performance, we plan to investigate integration with other neural network models.

CRediT authorship contribution statement

Ming Chen: Conceptualization of this study, Project administration, Methodology. **Yunhao Li:** Investigation, Software. **Xiuze Zhou:** 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 thank Michael McAllister for proofreading this paper.

References

[1] B.R. Smith, G. Linden, Two decades of recommender systems at Amazon.com, IEEE Internet Comput. 21 (3) (2017) 12–18.

[2] H. Zhu, X. Li, P. Zhang, G. Li, J. He, H. Li, K. Gai, Learning tree-based deep model for recommender systems, in: The 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018, pp. 1079–1088.

[3] P. Covington, J. Adams, E. Sargin, Deep neural networks for youtube recommendations, in: The 10th ACM Conference on Recommender Systems, 2016, pp. 191–198.

[4] Y. Jin, N. Tintarev, K. Verbert, Effects of personal characteristics on music recommender systems with different levels of controllability, in: The 12th ACM Conference on Recommender Systems, 2018, pp. 13–21.

[5] X. Yang, Y. Guo, Y. Liu, H. Steck, A survey of collaborative filtering based social recommender systems, Comput. Commun. 41 (2014) 1–10.

[6] W. Fan, Y. Ma, D. Yin, J. Wang, J. Tang, Q. Li, Deep social collaborative filtering, in: The 13th ACM Conference on Recommender Systems, 2019, pp. 305–313.

[7] H. Wang, N. Shao, D. Lian, Adversarial binary collaborative filtering for implicit feedback, in: The 33rd AAAI Conference on Artificial Intelligence, 2019, pp. 5248–5255.

[8] L. Zou, L. Xia, Y. Gu, X. Zhao, D. Yin, Neural interactive collaborative filtering, in: The 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 749–758.

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

[10] X. Zhou, S. Wu, Rating LDA model for collaborative filtering, Knowl.-Based Syst. 110 (01) (2016) 135–143.

[11] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T. Chua, Neural collaborative filtering, in: The 26th International Conference on World Wide Web, 2017, pp. 173–182.

[12] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, T. Chua, Attentional factorization machines: Learning the weight of feature interactions via attention networks, in: The 26th International Joint Conference on Artificial Intelligence, 2017, pp. 3119–3125.

[13] X. He, X. Du, X. Wang, F. Tian, J. Tang, T. Chua, Outer product-based neural collaborative filtering, in: The 27th International Joint Conference on Artificial Intelligence, 2018, pp. 2227–2233.

[14] W. Yu, H. Zhang, X. He, X. Chen, L. Xiong, Z. Qin, Aesthetic-based clothing recommendation, in: The 27th International Conference on World Wide Web, 2018, pp. 649–658.

[15] D. Liang, R.G. Krishnan, M.D. Hoffman, T. Jebara, Variational autoencoders for collaborative filtering, in: The 27th International Conference on World Wide Web, 2018, pp. 689–698.

[16] X. Li, J. She, Collaborative variational autoencoder for recommender systems, in: The 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 305–314.

[17] Y. Wu, C. Dubois, A.X. Zheng, M. Ester, Collaborative denoising autoencoders for top-n recommender systems, in: The 9th ACM International Conference on Web Search and Data Mining, 2016, pp. 153–162.

[18] H. Wang, N. Wang, D. Yeung, Collaborative deep learning for recommender systems, in: The 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 1235–1244.

[19] F. Strub, J. Mary, Collaborative filtering with stacked denoising autoencoders and sparse inputs, in: The 29th NIPS Workshop on Machine Learning for ECommerce, 2015.

[20] S. Lu, H. Chen, X. Zhou, B. Wang, H. Wang, Q. Hong, Graph-based collaborative filtering with MLP, Math. Probl. Eng. 2018 (2018).

[21] D. Liang, J. Altosaar, L. Charlin, D.M. Blei, Factorization meets the item embedding: regularizing matrix factorization with item co-occurrence, in: The 10th ACM Conference on Recommender Systems, 2016, pp. 59–66.

[22] B. Song, X. Yang, Y. Cao, C. Xu, Neural collaborative ranking, in: The 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 1353–1362.

[23] D. Cao, X. He, L. Miao, Y. An, C. Yang, R. Hong, Attentive group recommendation, in: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, pp. 645–654.

[24] J. Chen, F. Zhuang, X. Hong, X. Ao, Q. He, Attention-driven factor model for explainable personalized recommendation, in: The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, 2018, pp. 909–912.

[25] M. Chen, X. Zhou, DeepRank: Learning to rank with neural networks for recommendation, Knowl.-Based Syst. 209 (2020) 106478.

[26] X. Wang, R. Wang, C. Shi, G. Song, Q. Li, Multi-component graph convolutional collaborative filtering, in: The 34th AAAI Conference on Artificial Intelligence, 2020.

[27] F. Harper, J. Konstan, The movielens datasets, ACM Trans. Interact. Intell. Syst. 5 (2015) 1–19, http://dx.doi.org/10.1145/2827872.

[28] B.M. Sarwar, G. Karypis, J.A. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: The 10th International Conference on World Wide Web, 2001, pp. 285–295.

[29] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidtthieme, BPR: Bayesian personalized ranking from implicit feedback, in: The 25th Conference on Uncertainty in Artificial Intelligence, 2012, pp. 452–461.

[30] Z. Deng, L. Huang, C. Wang, J. Lai, P.S. Yu, Deepcf: A unified framework of representation learning and matching function learning in recommender system, in: The 33rd AAAI Conference on Artificial Intelligence, 2019, pp. 61–68.

[31] S. Rendle, W. Krichene, L. Zhang, J. Anderson, Neural collaborative filtering vs. matrix factorization revisited, 2020, arXiv preprint arXiv:2005.09683.

[32] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, Comput. Sci. (2014).

[33] H. Xue, X. Dai, J. Zhang, S. Huang, J. Chen, Deep matrix factorization models for recommender systems, in: The 26th International Joint Conference on Artificial Intelligence, 2017, pp. 3203–3209.

Chen Ming received the Ph.D. degree in Computer Science and Technology from Zhejiang University. He is currently working as Chief Technology Officer in Hithink RoyalFlush Information Network Co., Ltd. & Zhejiang Hithink RoyalFlush Artificial Intelligence Research Institute. He currently focuses on Big Data, Machine Learning, and Computer Vision. He has rich practical experience in the application of AI in financial field and medical field.

Yunhao Li born in Chanchun, Jilin Province, China. Being a senior student in Harbin Institute of Technology and exchanged in Johns Hopkins University during 2019 to 2020. Interested in Machine Learning and Data Analysis.

Xiuze Zhou received the M.S. degree from Xiamen University, in 2016. He is currently a researcher with the AI Research Institute, Zhejiang Hithink RoyalFlush, China. His current research interests include Machine Learning, Deep Learning and Recommender Systems.