Deep Item-based Collaborative Filtering for Top-N Recommendation

FENG XUE, Hefei University of Technology, China
XIANGNAN HE, National University of Singapore, Singapore
XIANG WANG, National University of Singapore, Singapore
JIANDONG XU, Hefei University of Technology, China
KAI LIU, Hefei University of Technology, China
RICHANG HONG, Hefei University of Technology, China

Item-based Collaborative Filtering (short for ICF) has been widely adopted in recommender systems in industry, owing to its strength in user interest modeling and ease in online personalization. By constructing a user's profile with the items that the user has consumed, ICF recommends items that are similar to the user's profile. With the prevalence of machine learning in recent years, significant processes have been made for ICF by learning item similarity (or representation) from data. Nevertheless, we argue that most existing works have only considered linear and shallow relationship between items, which are insufficient to capture the complicated decision-making process of users.

In this work, we propose a more expressive ICF solution by accounting for the nonlinear and higher-order relationship among items. Going beyond modeling only the second-order interaction (*e.g.,* similarity) between two items, we additionally consider the interaction among all interacted item pairs by using nonlinear neural networks. Through this way, we can effectively model the higher-order relationship among items, capturing more complicated effects in user decision-making. For example, it can differentiate which historical itemsets in a user's profile are more important in affecting the user to make a purchase decision on an item. We treat this solution as a deep variant of ICF, thus term it as DeepICF. To justify our proposal, we perform empirical studies on two public datasets from MovieLens and Pinterest. Extensive experiments verify the highly positive effect of higher-order item interaction modeling with nonlinear neural networks. Moreover, we demonstrate that by more fine-grained second-order interaction modeling with attention network, the performance of our DeepICF method can be further improved.

CCS Concepts: • **Information systems → Recommender systems**;

Authors' addresses: F. Xue, J. Xu, K. Liu and R. Hong, No. 193, Tunxi Road, Hefei, Anhui Province, P.R. China 230009; emails: feng.xue@hfut.edu.cn, jdxu2015@mail.hfut.edu.cn, kliu2017@mail.hfut.edu.cn, hongrc@hfut.edu.cn; X. Wang and X. He, Lab for Media Search, School of Computing, National University of Singapore, 13 Computing Drive, Singapore 117417; emails: xiangwang@u.nus.edu, xiangnanhe@gmail.com.
The corresponding author is Xiangnan He.
Authors' addresses: Feng Xue, feng.xue@hfut.edu.cn, Hefei University of Technology, School of Computer and Information, 193 Tunxi Road, Hefei, Anhui Province, 230009, China; Xiangnan He, xiangnanhe@gmail.com, National University of Singapore, School of Computing, 13 Computing Drive, 117417, Singapore; Xiang Wang, xiangwang@u.nus.edu, National University of Singapore, School of Computing, 13 Computing Drive, 117417, Singapore; Jiandong Xu, jdxu2015@mail.hfut.edu.cn, Hefei University of Technology, School of Computer and Information, 193 Tunxi Road, Hefei, Anhui Province, 230009, China; Kai Liu, kliu2017@mail.hfut.edu.cn, Hefei University of Technology, School of Computer and Information, 193 Tunxi Road, Hefei, Anhui Province, 230009, China; Richang Hong, hongrc@hfut.edu.cn, Hefei University of Technology, School of Computer and Information, 193 Tunxi Road, Hefei, Anhui Province, 230009, China.

XXXX-XXXX/2018/11-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Additional Key Words and Phrases: Collaborative Filtering, Item-based CF, Neural Networks, Deep Learning, Implicit Feedback

ACM Reference Format:
Feng Xue, Xiangnan He, Xiang Wang, Jiandong Xu, Kai Liu, and Richang Hong. 2018. Deep Item-based Collaborative Filtering for Top-N Recommendation. *Submitted to ACM Transactions on Information Systems* x, x (November 2018), 25 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In the era of information overload, recommender systems play a pivotal role in many user-oriented online services such as E-commerce, content-sharing sites, and news portal. An effective recommender system not only can facilitate the information seeking process of users, but also can create customer loyalty and increase profit for the company. With such an important role in online information systems, recommendation has become an active topic of research and attracted increasing attention in information retrieval and data mining communities [16, 40, 44].

Among various recommendation strategies, collaborative filtering (CF) is now the dominant one and has been widely adopted in industry [25, 36]. By leveraging user-item interaction data to predict user preference, CF is mostly used in the candidate selection phase of a recommender system [41], which is complemented by an integrated ranking engine that integrates various signal to rank the candidates selected by CF. Generally speaking, CF techniques can be divided into two types — user-based and item-based approaches. The *matrix factorization* (MF) model [17] is a representative user-based CF method (short for UCF), which represents a user with an ID and projects the ID into the same embedding space of items; then the relevance score between a user-item pair is estimated as the inner product of the user embedding and item embedding. In contrast, item-based CF (short for ICF) represents a user with her historically interacted items, using the similarity between the target item and interacted items to estimate the user-item relevance [15, 36].

1.1 Why Item-based Collaborative Filtering?

Despite the popularity of MF in recommendation research, there are several advantages of ICF over UCF. First, by representing a user with her consumed items, ICF encodes more signal in its input than UCF that simply uses an ID to represent a user. This provides ICF more potential to improve both the accuracy [9] and interpretability [36] of user preference modeling. For example, there are several empirical evidences on accuracy superiority of ICF over UCF methods for top-N recommendation [8, 9, 42]; and ICF can interpret a recommended item as its high similarity with some items that the user has consumed before, which would be more acceptable by users than "similar users" based explanation scheme [46]. Second, the composability of ICF in user preference modeling makes it easier to implement online personalization [15]. For example, when a user has new purchases, instead of re-training model parameters to refresh recommendation list, ICF can approximate the refreshed list by simply retrieving items that are similar to the new purchased items. Such strategy has successfully provided instant personalization in YouTube based on user recent watches (*cf.* Section 6.2.3 Instant Recommendation of [3]). By contrast, UCF methods like MF associate model parameters with an user ID, making them compulsory to update model parameters to refresh the recommendation list for a user (*cf.* the online-update strategy for MF [17, 30]).

Early ICF approaches use statistical measures such as Pearson correlation and cosine similarity to quantify the similarity between two items [32]. However, such methods typically require extensive manual tuning on the similarity measure to make them perform well, and it is non-trivial to adapt a well-tuned method to a new dataset or dataset of a new product domain. In recent years, data-driven methods have been developed to learn item similarity from data, among which two representative methods are the *sparse linear method* (SLIM) [27] and *factored item similarity model*

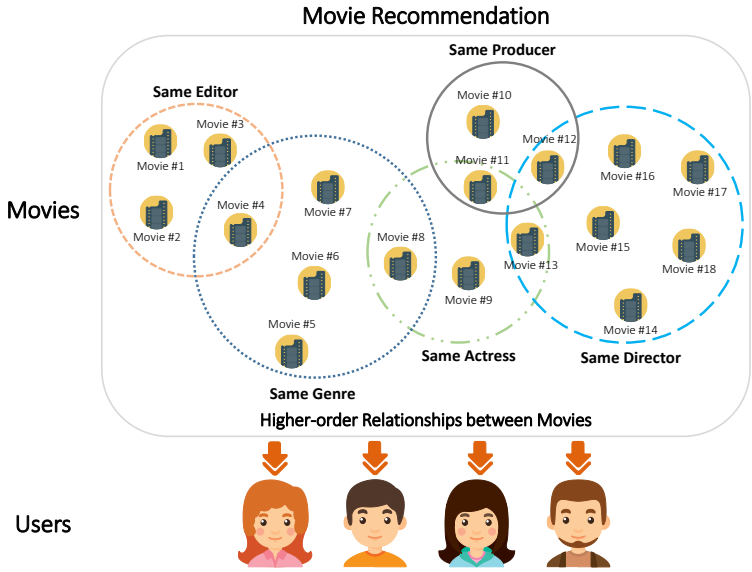


Fig. 1. An illustrative example of higher-order item relations in the Movie domain. Besides such explicit relations based on item attributes, there are possibly implicit higher-order relations, such as an itemset that is frequently co-purchased by users due to functional complementarity (*e.g.,* mouse, keyboard, and screen).

(FISM) [19]. In SLIM, the item-item similarity matrix is directly learned with additional constraints on sparsity and non-negativity; in FISM, the similarity between two items is factorized as the inner product of their latent vectors (*aka.* embeddings), which can be seen as assuming the item-item similarity matrix to be low-rank. Some recent developments for ICF include the *neural attentive item similarity* (NAIS) model [15] which extends FISM by using attention network to discriminate which item-item similarities are more important for a prediction, the *collaborative denoising auto-encoder* (CDAE) [42] which uses nonlinear auto-encoder architecture [33] to learn item similarity, and the *global and local SLIM* (GLSLIM) [9] which uses different SLIM models for different user subsets.

1.2 Why Higher-Order Item Relations?

However, we argue that these previous efforts on ICF only model the second-order relations between pairs of items, more specifically, the relation between an item in user history and the target item. Higher-order relations, such as multiple items that share certain properties which are likely to be consumed together, are not considered. Figure 1 illustrates some high-order relations among movies, such as directed by the same director, acted by the same actress, produced by the same producer, and so on. Such relations can be even more complicated when considering the possible overlap among multiple relations, such as a user subset likes to watch movies that share the same director and actress (*e.g.,* Movie #12 and Movie #13 in Figure 1). Besides such explicit high-order relations based on item attributes, some implicit relations may also exist. For example, a set of items may be frequently bought together by users, since they are complementary in functionality (*e.g.,* mouse, keyboard, and screen) or even they have no interpretable reason. We believe that such higher-order item relations provide valuable signal to estimate user preference, and such that, the recommendation accuracy of ICF can be significantly improved if such higher-order relations can be properly taken into account.

Essentially, Christakopoulou and Karypis [8] have verified the existence of higher-order item relations on several product domains and demonstrated their effectiveness in ICF. Nevertheless,

we argue that their proposed *higher-order sparse linear method* (HOSLIM) is limited in integrating higher-order relations in a static and linear manner. Specifically, they first identify frequent itemsets from user-item interaction data, and then extend SLIM to learn the *item-itemset similarity matrix*, which is then used to capture the higher-order item relations. One difficulty of such two-step solution is that, the identification of frequent itemsets requires a support threshold, which needs to be carefully tuned to avoid negative effects. If an itemset is useful but fails to be identified in the first step, the following predictive model cannot capture its impact; meanwhile, if an itemset is useless but is identified in the first step, it will have an uncontrollable impact on the predictive model which is unexpected. As such, a unified ICF solution that can automatically encode the impact of higher-order item relations into user preference modeling and prediction is highly desired.

1.3 Our Proposal and Contributions

In this work, we aim to fill the research gap of developing ICF models that can effectively capture higher-order item relations. We leverage on the recent success of neural recommender models [16] and develop a neural network method to achieve the target. Distinct to HOSLIM [8] that detects higher-order item relations in a separate step, we integrate the learning of higher-order item relations into the predictive model that captures second-order item relations, but use different neural components to capture the two kinds of item relations. Specifically, in the low-level of the neural network, we first model second-order item relations via a multiply operation on each pair of item embeddings (similar to the setting of FISM [19] and NAIS [15]); above the pairwise interaction layer, we then stack multiple layers to learn higher-order item relations in a non-linear way. Owing to the strong function learning ability of multi-layer neural network, this end-to-end solution is expected to capture the complicated impacts of higher-order item relations in user decision-making. Since this solution can be treated as a deep variant of ICF under the context of neural network modeling, we term it as DeepICF. We conduct extensive experiments on two datasets from MovieLens and Pinterest, verifying the highly positive effect of higher-order item relation modeling in DeepICF. Moreover, we integrate the attention design in the recently proposed NAIS [15] to refine the modeling of second-order item relations (*i.e.,* pairwise item similarities), which leads to further improvements.

The key contributions of this work are outlined as follows.

- A generic neural network framework is proposed to model higher-order item relations for item-based CF. The key idea is simple in using multiple nonlinear layers above the pairwise interaction modeling to learn higher-order item relations.
- Two specific methods under the framework which differ in the pairwise item relation modeling are presented. One method (DeepICF) combines pairwise item interactions with the same weight and the other method (DeepICF+a) uses an attention mechanism to differentiate the importance of pairwise interactions.
- Extensive experiments are performed on two real-world datasets to verify the effectiveness of our proposal. Codes have been released to facilitate further developments on deep item-based CF methods: https://github.com/linzh92/DeepICF.

The rest of this paper is organized as follows. We first provide some preliminaries for ICF in Section 2. We then elaborate our proposed DeepICF methods in Section 3. Afterwards we report experimental results in 4 and review related work in Section 5. Finally we conclude the paper and highlight some future directions in Section 6.

2 PRELIMINARIES

We first brief the general framework for item-based collaborative filtering. We then discuss the Higher-Order SLIM method, which is an existing solution to model higher-order item relations for ICF. Lastly, we recapitulate the FISM and NAIS methods, which are representation learning-based ICF methods that form the basis of our DeepICF methods.

2.1 Framework of Item-based Collaborative Filtering

ICF predicts a user-item interaction y_{ui} by assuming that user u's preference on item i depends on the similarity of i to all items that u has interacted with before. In general, the predictive model of ICF can be abstracted as,

$$\hat{y}_{ui} = \sum_{j \in \mathcal{R}_u^+} s_{ij} r_{uj}, \tag{1}$$

where $\mathcal{R}_u^+$ is the item set that the user u has interacted with, s_{ij} denotes the similarity between item i and j, and r_{uj} is the u's observed preference on item j, which can be a real-valued rating score (explicit feedback) and a binary 0 or 1 (implicit feedback).

We summarize the advantages of ICF over UCF in threefold: accuracy, interpretability, and ease on online recommendation. By **accuracy**, it is arguable that characterizing a user with her interacted items in ICF is can capture the user's interest in a more explicit way. In contrast, in UCF, a static set of parameters to describe a user (*e.g.,* user embedding in MF) has limited representation power in reflecting the dynamic and evolving user preference. Moreover, several prior efforts [8, 9, 42] provide empirical evidences on accuracy superiority of ICF over UCF. By **interpretability**, ICF can interpret why a recommendation is made via the explanation mechanism: because the item is similar to which item you liked before. Such explanation is more concrete and more acceptable by users than "because similar users also like it" provided by UCF [46]. By **ease on online recommendation**, the composability of ICF in user preference modeling —*i.e.,* sum over the item similarities — makes it more suitable for online recommendation. Particularly, when a user has new purchases, UCF needs to update model parameters, *e.g.,* user embeddings, to refresh the recommendation list, which is difficult to be adopted for real-time personalization. In contrast, based on the offline item similarities, ICF can approximate the refreshed list by simply retrieving items that are similar to the new purchased ones.

Clearly, the estimation of item similarity s_{ij} is crucial to the performance of ICF. A straightforward solution is to employ statistical measures, such as Pearson correlation and cosine similarity, on item features to estimate it. Recently, date-driven methods have been developed to learn the item similarity from data, which better tailor the similarity parameters for the specific dataset.

2.2 SLIM and Higher-Order SLIM

SLIM [27] is among the first learning-based ICF methods that direct learn the item-item similarity matrix from historical interaction data. Specially, it minimizes the reconstruction error between the original user-item interaction matrix and the reconstructed one that is derived from an ICF model. Two constraints are employed on the item-item similarity matrix: no-negativity and sparsity, which ensure the meaningfulness of the learned similarities and enforce each item to be similar to a few items only. The objective function of SLIM is formulated as,

$$\mathcal{L}_{SLIM} = \frac{1}{2} \sum_{u=1}^{U} \sum_{i=1}^{I} (r_{ui} - \sum_{j \in \mathcal{R}_u^+} s_{ij} r_{uj})^2 + \lambda_1 \left\| \mathbf{S} \right\|_1 + \lambda_2 \left\| \mathbf{S} \right\|_2, \tag{2}$$
$$subject\ to\ \mathbf{S} \geq 0, diag(\mathbf{S}) = 0,$$

where U and I denote the number of users and items, $\mathbf{S} \in \mathbb{R}^{I \times I}$ represents the item-item similarity matrix where each entry is s_{ij}, λ_1 is a hyper-parameter to control the strength of L_1 regularization to enforce the sparsity constraint, and λ_2 is a hyper-parameter to control the strength of L_2 regularization to prevent overfitting. In SLIM, item similarity matrix $\mathbf{S}$ is the parameter to learn — the constraint $\mathbf{S} \geq 0$ is performed to ensure each element in $\mathbf{S}$ (*i.e.,* a similarity score) to be non-negative, and the constraint $diag(\mathbf{S}) = 0$ forces the diagonal elements of $\mathbf{S}$ to be zero to eleminate the impact the target item itself in estimating a prediction.

Despite effectiveness, the expressiveness of SLIM can be limited by its modeling of pairwise item relations only, since it overlooks the possible higher-order relations, such as multiple items belong the same group, share the same attributes, co-occur frequently, and so on. To this end, Christakopoulou and Karypis [8] propose HOSLIM which extends SLIM to capture higher-order item relations. In particular, they first apply frequent itemset mining algorithm to identify itemsets that are frequently co-interacted by users; they then extend SLIM to jointly learn the item-item similarities and itemset-item similarity, which can capture higher-order item relations. Specifically, the predictive model of HOSLIM is as follows,

$$\hat{y}_{ui} = \mathbf{r}_u^\top \mathbf{s}_i + \mathbf{r'}_u^\top \mathbf{s}'_i, \tag{3}$$

where $\mathbf{r}_u \in \mathbb{R}^I$ denotes the interaction vector of u on items, and $\mathbf{r}'_u \in \mathbb{R}^{I'}$ denotes the interaction vector of u on itemsets (I' itemsets in total), where each entry $r'_{ui'}$ denotes whether u has interacted with all items in itemset i'. Vectors $\mathbf{s}_i$ and $\mathbf{s}'_i$ are model parameters to learn, where $\mathbf{s}_i \in \mathbb{R}^I$ denotes the similarity vector of i on items, and $\mathbf{s}'_i \in \mathbb{R}^{I'}$ denotes the similarity vector of i on itemsets. The objective function of HOSLIM is similar to that of SLIM with additional constraints on the itemset similarity matrix:

$$\mathcal{L}_{HOSLIM} = \frac{1}{2} \sum_{u=1}^{U} \sum_{i=1}^{I} (r_{ui} - \hat{y}_{ui})^2 + \lambda_1 (\|\mathbf{S}\|_1 + \|\mathbf{S}'\|_1) + \lambda_2 (\|\mathbf{S}\|_2^2 + \|\mathbf{S}'\|_2^2), \tag{4}$$

$$subject\ to\ \mathbf{S} \geq 0,\ \mathbf{S}' \geq 0,\ s_{ii} = 0,\ s'_{ij} = 0\ where\ \{j \in \mathcal{I}_i\}, \tag{5}$$

where $\mathbf{S} \in \mathbb{R}^{I \times I}$ and $\mathbf{S}' \in \mathbb{R}^{I \times I'}$ denote the item-item and item-itemset similarity matrix (of which each vector is $\mathbf{s}_i$ and $\mathbf{s}'_i$), respectively. $\mathcal{I}_i$ denotes the itemsets that contain item i. The definition of the constraints follows the same logic of SLIM thus we omit the explanation here.

As mentioned in introduction, the two-step solution of HOSLIM has several limitations. First, as we can see that from Equation (3), the user-itemset interaction vector $\mathbf{r}'_u$ plays an important role in the predictive model; however, it is determined by the frequent itemset mining algorithm, which requires a support threshold that is non-trivial to tune for different datasets (since the item frequency distribution of different datasets may vary a lot). Second, in HOSLIM, higher-order item relation is defined as the similarity between an itemset and an item, which is aggregated in the same way as that of second-order item similarities — *i.e.,* linear and static — in the predictive model. By linear, we mean that the similarities between candidate itemsets and target item are directly summed in the predictive model; by static, we mean that for different predictions, the importance of itemset-item similarities remain the same (*i.e.,* a uniform weight of 1). Such a simple manner to model higher-order item relations omits the varying importance of itemsets for a prediction and the possible nonlinear relations among items, making it suboptimal to predict user preference. Moreover, this implies the difficulty of modeling higher-order relations in traditional linear recommendation models, revealing the necessity and possibility of addressing it with nonlinear, more expressive, and end-to-end trainable neural network models. This forms the major motivation of this work from the technical perspective.

2.3 FISM and NAIS Methods

FISM [19] stands for another mainstream in learning-based ICF — instead of directly learning the whole item similarity matrix which can be very space- and time- consuming, it applies a low-rank assumption on the item similarity matrix and learns the low-rank structure to reconstruct the matrix. The predictive model of FISM is formulated as,

$$\hat{y}_{ui} = \frac{1}{(|\mathcal{R}_u^+| - 1)^\alpha} \sum_{j \in \mathcal{R}_u^+ \setminus \{i\}} r_{uj} \underbrace{\mathbf{p}_i^\top \mathbf{q}_j}_{s_{ij}}, \tag{6}$$

where $\mathbf{p}_i \in \mathbb{R}^k$ and $\mathbf{q}_j \in \mathbb{R}^k$ are trainable parameters that define the low-rank structure and k denotes the rank size; from the perspective of representation learning, $\mathbf{p}_i$ and $\mathbf{q}_j$ can be seen as the latent features (*aka.* embedding) for target item i and historical item j. As can be seen, the item similarity score s_{ij} can be expressed as the inner product between $\mathbf{p}_i$ and $\mathbf{q}_j$. Hyper-parameter α controls the normalization on users of different history length, *e.g.*, $\alpha = 0$ means no normalization is used, $\alpha = 1$ means L_1 normalization is used, and other intermediate values between 0 and 1 are also applicable. Following the zero diagonals constraint in SLIM, the sum over item set $\mathcal{R}_u^+ \setminus \{i\}$ is to exclude the influence of the target item i in constructing u's profile to predict $\hat{y}_{ui}$, which can avoid information leak during training. Moreover, since most recommender systems deal with implicit feedback where $r_{uj} = 1$ for all observed interactions, we can omit the coefficient r_{uj} in Equation (6).

It is arguable that FISM is limited since it models all second-order item relations with a same weight for all predictions. To address this limitation, NAIS is proposed very recently [15] which applies a dynamic weighting strategy on second-order item relations. Specifically, NAIS considers that the historical items of u should have different contributions on the prediction of $\hat{y}_{ui}$. An attention network is then employed to learn the varying weights of item-item relations based on item embeddings. The predictive model of NAIS is formulated as,

$$\hat{y}_{ui} = \sum_{j \in \mathcal{R}_u^+ \setminus \{i\}} a_{ij} \mathbf{p}_i^\top \mathbf{q}_j, \tag{7}$$

where a_{ij} denotes the attentive weight of similarity s_{ij} in contributing to the final prediction. In NAIS, a_{ij} is parameterized as a neural network's output with item embeddings $\mathbf{p}_i$ and $\mathbf{q}_j$ as input. Specifically, two neural attention networks are presented which differ in how to combine $\mathbf{p}_i$ and $\mathbf{q}_j$:

$$\begin{cases} a_{ij} = \text{softmax}'(f(\mathbf{p}_i, \mathbf{q}_j)) \\ f_{concat(\mathbf{p}_i, \mathbf{q}_j)} = \mathbf{h}^\top ReLU(\mathbf{W}[\mathbf{p}_i, \mathbf{q}_j] + \mathbf{b}) \\ f_{prod(\mathbf{p}_i, \mathbf{q}_j)} = \mathbf{h}^\top ReLU(\mathbf{W}(\mathbf{p}_i \odot \mathbf{q}_j) + \mathbf{b}) \end{cases}, \tag{8}$$

where softmax′ is a variant of the softmax function that takes the normalization on user history length into account. As such, the normalization term $\frac{1}{(|\mathcal{R}_u^+|-1)^\alpha}$ in FISM can be omitted in NAIS. $\mathbf{W}$ and $\mathbf{b}$ are the weight matrix and bias vector of the hidden layer of the attention network, and $\mathbf{h}$ is the weight vector that projects the hidden layer into the scalar output.

While FISM and NAIS have provided strong performance for item recommendation, we argue that neither of them takes the higher-order item relations into account. When certain higher-order item relations exist in the data, as have demonstrated in [8] on several real-world datasets, both methods cannot capture them and thus may provide suboptimal performance. In next section, we present our neural network modeling approach that specifically accounts for higher-order item relations for user preference prediction.

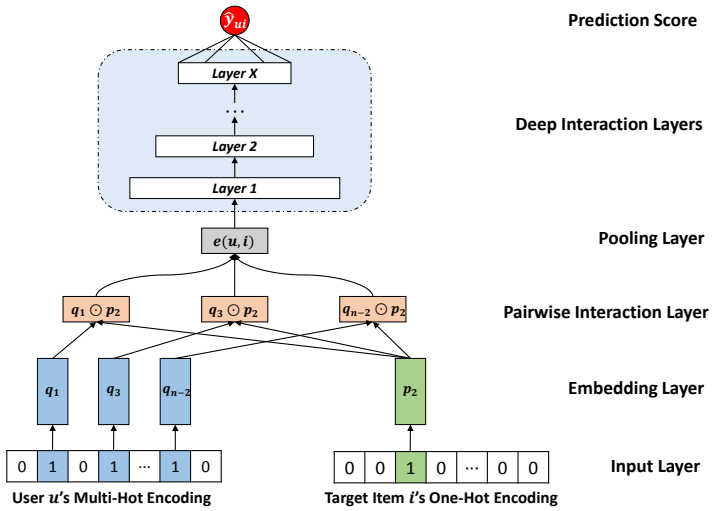


Fig. 2. Our proposed neural network architecture of DeepICF which captures both second-order and higher-order interactions among items. Given a user-item pair (u, i) where the user is expressed as u's historically interacted items, the neural network outputs $\hat{y}_{ui}$ denoting the estimated preference of user u on item i.

3 METHODS

This section elaborates our proposed methods. In Section 3.1, we first discuss the predictive model, *i.e.,* given a user-item pair (u, i) how to estimate the prediction value $\hat{y}_{ui}$. Specifically, we first present a general framework for higher-order item relation modeling with neural networks (*cf.* Figure 2), and then discuss two instantiations under the framework — *DeepICF* that uses standard average pooling on second-order interactions, and *DeepICF+a* that uses an adaptive pooling strategy with attention on second-order interactions (*cf.* Figure 4). In Section 3.2, we describe the learning procedure of the models. Lastly in Section 3.4, we discuss the connections of our methods with existing models, shedding lights on the rationality of our proposed methods analytically.

3.1 Model

Figure 2 illustrates our proposed framework to model higher-order item relations for ICF. The overall neural network architecture follows the design of the *neural collaborative filtering* (NCF) framework [16] with two major differences. First, in the input layer that represents a user (bottom left), distinct from NCF that applies one-hot encoding on the user's ID, we use multi-hot encoding on the user's interacted items. This naturally leads to the difference in the embedding layer — rather than using a vector to represent the user, we use a set of vectors where each vector represents an interacted item of the user. Second, instead of designing a holistic neural CF layers to model the interaction between user and item, we divide the interaction modeling into two components — 1) pairwise interaction layer that models the interaction between each historical item and the target item, and 2) deep interaction layers that model higher-order interaction among all historical items and the target item. Next, we elaborate the architecture layer by layer.

Input and Embedding Layer. In the right channel that represents the target item i, one-hot encoding on the ID feature of i is applied. Then the ID is projected to an embedding vector $\mathbf{p}_i \in \mathbb{R}^k$ to describe the target item where k denotes the embedding size. In the left channel that represents the user u, multi-hot encoding on the ID feature of u's interacted items $\mathcal{R}_u^+$ is applied. Then for

each historical item $j \in \mathcal{R}_u^+$, we project it to an embedding vector $\mathbf{q}_j \in \mathbb{R}^k$. As such, the output of the embedding layer is a set of vectors $\mathcal{Q}_u = \{\mathbf{q}_j | j \in \mathcal{R}_u^+\}$ that represents the user u and a vector $\mathbf{p}_i$ that represents the target item i.

Note that another way to represent target item is to base on the users that have interacted with it (as used in the *deep matrix factorization* model [43]), which is arguably more informative but costly. Since this work is focused on ICF that aims to exploit item relations, we leave this exploration as future work.

Besides ID, the input of embedding layer can be easily extended to incorporate side information, such as location, time, and item attributes. To be specific, each feature can be mapped to an ID via one-hot encoding. Thereafter, we feed them into the embedding layer to establish its embeddings, which are scaled by the feature value — 1 for discrete features (*e.g.,* , user gender and item attributes) and real value for numerical features (e.g., click number). Since the work focuses on the pure collaborative filtering setting, we leave the incorporation of side information as future work.

Pairwise Interaction Layer. Inspired by the effectiveness of FISM and NAIS, we apply the similar way to explicitly model the interaction between each historical item and target item. Specifically, we apply element-wise product on their embedding vectors, obtaining a set of pairwise interacted vectors $\mathcal{V}_{ui} = \{\mathbf{q}_j \odot \mathbf{p}_i | j \in \mathcal{R}_u^+ \backslash i\}$ which capture the second-order relations between the target item i and u's historically interacted items.

Theoretically speaking, other than element-wise product, any binary function can be applied here to map $\mathbf{q}_j$ and $\mathbf{p}_i$ to one vector that encodes their interaction. For example, addition ($\mathbf{q}_j + \mathbf{p}_i$), subtraction ($\mathbf{q}_j - \mathbf{p}_i$), division ($\frac{\mathbf{q}_j}{\mathbf{p}_i + \sigma}$), and so on. Here we choose element-wise product mainly because of its generalizing of inner product to vector space [14], which can sufficiently capture the signal in inner product. In FISM, the use of inner product to capture second-order item interactions implies that the item similarity matrix has a low-rank structure, which leads to good estimation on item similarities. As such, the output vectors of this layer $\mathcal{V}_{ui}$ are supposed to encode the signal of pairwise item similarities.

Pooling Layer. Since the number of historical items of different users may vary, the output of pairwise interaction layer will have different sizes. The pooling layer operates on the vectors in variable-size $\mathcal{V}_{ui}$, aiming to produce a vector of fixed size to facilitate further processing. Here, we consider two choices — *weighted average pooling* and *attention-based pooling* — which lead to our two proposed methods *DeepICF* and *DeepICF+a*.

The weighted average pooling used in DeepICF is defined as follows,

$$f_{avg}(\mathcal{V}_{ui}) = \frac{1}{|\mathcal{V}_{ui}|^\alpha} \sum_{\mathbf{v} \in \mathcal{V}_{ui}} \mathbf{v} = \frac{1}{(|\mathcal{R}_u^+| - 1)^\alpha} (\sum_{j \in \mathcal{R}_u^+ \backslash i} \mathbf{q}_j \odot \mathbf{p}_i), \tag{9}$$

where α is the normalization hyper-parameter that controls the smoothing on $\mathcal{V}_{ui}$ of different sizes. When α is set to 1, no smoothing is used and it becomes the standard average pooling; when α is set to 0, the operation downgrades to the standard sum pooling. Since the distribution of user activity level may vary for different datasets, there is no uniformly optimal setting for α and it should be separately tuned for different datasets. However, regardless of the value of α, all historical items of a user will contribute equally to the prediction on all target items, which is an unrealistic assumption as argued in [15]. Typically there are only a few items a user interacted before will affect the user's decision on an item. For example, when a user decides whether to purchase a phone cover, it should be the phones that he purchased before have a larger impact, rather than cameras or clothing products. As such, when modeling the interaction between historical items and target item, non-uniform weights should be applied to the historical items. Besides, we have

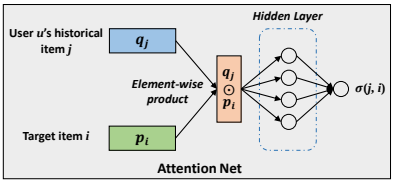


Fig. 3. Illustration of the attention network in DeepICF+a.

tried to assign a contribution weight w_j for each historical item $\mathbf{q}_j$ of user u, but the performance of such design is not significantly improved. As such, we do not further explore the extension.

The attention-based pooling used in DeepICF+a is designed to address the above-mentioned limitation in DeepICF. Inspired by the attention network design in ACF [6] and NAIS [15], we define the attention-based pooling as follows,

$$f_{att}(\mathcal{V}_{ui}) = \frac{1}{|\mathcal{V}_{ui}|^{\alpha}} \sum_{\mathbf{v} \in \mathcal{V}_{ui}} a(\mathbf{v}) \cdot \mathbf{v} = \frac{1}{(|\mathcal{R}_u^+| - 1)^{\alpha}} \sum_{j \in \mathcal{R}_u^+ \setminus i} a(\mathbf{q}_j \odot \mathbf{p}_i) \cdot (\mathbf{q}_j \odot \mathbf{p}_i), \tag{10}$$

where $a(\mathbf{v})$ is the attention function that takes vector $\mathbf{v}$ as input, and outputs the importance of $\mathbf{v}$ in the weighted average pooling. Figure 3 illustrates the structure of the attention network. Specifically, we use a multi-layer perceptron with a hidden layer to parameterize the attention function:

$$a(\mathbf{v}) = \text{softmax}'(\mathbf{h}^T ReLU(\mathbf{W}\mathbf{v} + \mathbf{b})) \tag{11}$$

where $\mathbf{W} \in \mathbb{R}^{k' \times k}$ and $\mathbf{b} \in \mathbb{R}^{k'}$ denote the weight matrix and bias vector of the attention network, respectively, and k' denotes the size of the hidden layer, which is also called as attention size. $\mathbf{h} \in \mathbb{R}^{k'}$ denotes the weights of the output layer of the attention network. softmax$'$ is a variant of the softmax function to normalize the attentive weights [15], defined as,

$$\text{softmax}'(a(\mathbf{v})) = \frac{\exp a(\mathbf{v})}{[\sum_{\mathbf{v} \in \mathcal{V}_{ui}} \exp a(\mathbf{v})]^{\beta}} \tag{12}$$

where β is a hyper-parameter to smooth the value of the denominator in softmax. Note that tuning β has a similar effect as tuning α, since both hyper-parameters can regulate the weights of second-order item interactions for users of different history lengths. In our experiments, we find that with a proper tuning on β (in the range of 0 to 1), setting α to 0 leads to satisfactory performance. Thus the normalization term $\frac{1}{(|\mathcal{R}_u^+|-1)^{\alpha}}$ can be omitted in DeepICF+a.

Deep Interaction Layers. The output of the previous pooling layer is a vector of dimension k, which condenses the second-order interaction between historical items and target item. Let the vector be $\mathbf{e}_{ui}$ (i.e., $\mathbf{e}_{ui} = f_{avg}(\mathcal{V}_{ui})$ for DeepICF and $\mathbf{e}_{ui} = f_{att}(\mathcal{V}_{ui})$ for DeepICF+a). Next we consider how to capture higher-order interactions among items on the basis of $\mathbf{e}_{ui}$. Inspired by our recent development on *neural factorization machines* (NFM) [14], we propose to stack a multi-layer perceptron (MLP) above $\mathbf{e}_{ui}$ to achieve the higher-order modeling. The rationality is quite similar — by treating the historical items $\mathcal{R}_u^+$ and target item i as features into NFM, the $\mathbf{e}_{ui}$ vector plays the same role as the output vector of the bi-interaction layer in NFM (i.e., encoding the similar semantics of pairwise interactions between feature embeddings). Analogously, the MLP above $\mathbf{e}_{ui}$ is capable of capturing higher-order interactions among feature embeddings. We refer interested readers to the NFM paper [14] and the Deep Crossing paper [34] on more analysis about how the use of MLP can capture higher-oder interactions among features.

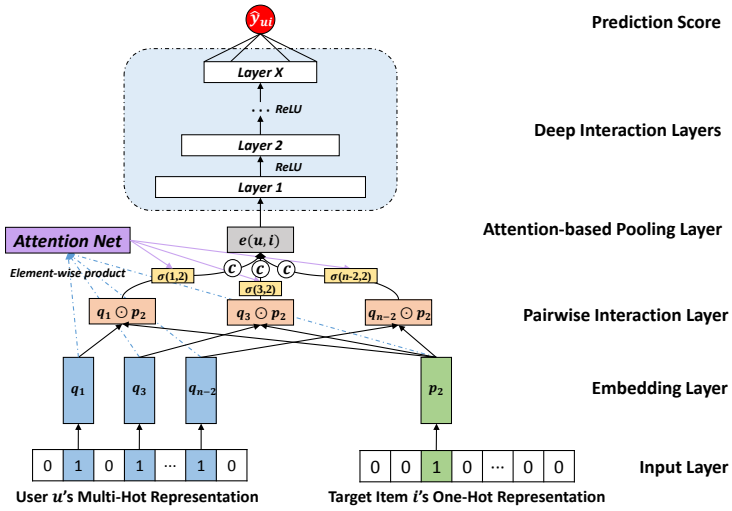


Fig. 4. Illustration of our attentional model DeepICF+a which introduces the attention mechanism to differentiate the varying contributions of user u's historically interacted items for the final prediction.

We give the formal definition of the deep interaction layers as follows,

$$\begin{cases} \mathbf{e}_1 = ReLU(\mathbf{W}_1\mathbf{e}_{ui} + \mathbf{b}_1) \\ \mathbf{e}_2 = ReLU(\mathbf{W}_2\mathbf{e}_1 + \mathbf{b}_2) \\ \dots \\ \mathbf{e}_L = ReLU(\mathbf{W}_L\mathbf{e}_{L-1} + \mathbf{b}_L) \end{cases}, \tag{13}$$

where $\mathbf{W}_l$, $\mathbf{b}_l$, and $\mathbf{e}_l$ denote the weight matrix, bias vector, activation function, and output vector of the l-th hidden layer, respectively. We use the rectifier unit as the activation function, which is known to be more resistant to the saturation issue when the network becomes deep, and empirically shows good performance in our setting. The size of each hidden layer is subjected to tune, and we adopt the conventional choice of tower structure. We will report the detailed setting and hyper-parameter tuning process in Section 4.1.

Prediction Layer. As the output of deep interaction layers, the vector $\mathbf{e}_L$ encodes informative prediction signal aggregated from second-order to higher-order item interactions. Since a multi-layer nonlinear network is able to fit any continuous function in theory [18], each dimension in $\mathbf{e}_L$ is supposed to encode the item interactions of any-order. We then project $\mathbf{e}_L$ to the prediction score with a simple linear regression model:

$$\hat{y}_{ui} = \mathbf{z}^\top \mathbf{e}_L + b_u + b_i, \tag{14}$$

where $\mathbf{z}$, b_u, and b_i denotes the weight vector, user bias, and item bias of the prediction layer, respectively. The two bias terms are to capture the variance in the popularity of different items and activity of different users, which were found to have an impact for learning from implicit feedback [19]. Each element in $\mathbf{z}$ measures the importance of the corresponding dimension in $\mathbf{e}_L$ for prediction. Here we make $\mathbf{z}$ a global parameter shared by all predictions. We note that a more fine-grained design is to tweak it be item-aware or user-aware or both, however it may also increase the model complexity and make the model more difficult to train. We leave this exploration as future work, since we find the current global setting leads to satisfactory performance.

3.2 Learning

Two mainstream methods for learning recommender models are to optimize pointwise [3, 16, 21] and pairwise [29, 40, 44] learning to rank objective functions. Focusing on implicit feedback, pointwise methods typically assign a predefined target value to observed user-item entries (i.e., positive examples) and sampled non-observed entries (i.e., negative examples), training model parameters to output similar values as the target values for both positive and negative examples. By contrast, pairwise methods assume that observed entries should have higher prediction scores than non-observed ones, performing optimization on the margin between positive and negative examples. To our knowledge and experience, there is no permanent winner between the two learning types and the performance depends largely on the predictive model and the dataset (see [19, 42] for more empirical evidence).

In this work, we opt for the pointwise log loss, which has been widely used for optimizing neural recommender models recently and demonstrated good performance [2, 16, 43]. It casts the learning as a binary classification task, minimizing the objective function as follows,

$$\mathcal{L} = \frac{-1}{|\mathcal{R}^+| + |\mathcal{R}^-|}[\sum_{(u,i)\in\mathcal{R}^+} \log \delta(\hat{y}_{ui}) + \sum_{(u,j)\in\mathcal{R}^-} \log(1 - \delta(\hat{y}_{uj}))] + \lambda \left\|\Theta\right\|^2, \tag{15}$$

where $\delta(\cdot)$ is the sigmoid function that restricts the prediction to be in $(0, 1)$. Set $\mathcal{R}^+$ denotes the positive examples, which are identical to observed user-item entries, and $\mathcal{R}^-$ denotes the set of negative examples, which are sampled from non-observed user-item entries. For each positive example (u, i), we sample NS negative examples (u, j) to pair with it, where NS is the negative sampling ratio. Consistent with previous findings on NCF models [16], we find the negative sampling ratio plays an important role on the performance of our DeepICF methods. A default setting of $NS = 4$ leads to good performance in most cases (empirical results are shown in Figure 10 in Section 4). Hyper-parameter λ controls the strength of L_2 regularization on model parameters Θ to prevent overfitting. Due to the use of fully connected MLP, the deep interaction layers in DeepICF methods are prone to overfitting. Thus, we mainly tune λ for the weight matrices $\{\mathbf{W}_1, \mathbf{W}_2, ..., \mathbf{W}_L\}$ in the deep interaction layers.

Pre-training. Due to the non-linearity of deep neural network models and the non-convexity of the learning problem, gradient descent methods can be easily trapped to local optimum solutions which are suboptimal. As such, model initialization plays an important role on the model's generalization performance [11]. We empirically find that our models suffer from slow convergence and poor performance when all model parameters are initialized randomly. To address the optimization difficulties and fully explore the potential of DeepICF models, we pre-train them with FISM. Specifically, we use the item embedding vectors learned by FISM to initialize the embedding layer of both DeepICF models. With such a meaningful initialization on the embedding layer, the convergence and performance of DeepICF can be greatly improved, even when other parameters are randomly initialized with a Gaussian distribution.

3.3 Time Complexity Analysis

In this subsection, we analyze the time complexity of our models of DeepICF and DeepICF+a. This directly reflects the time cost of DeepICF and DeepICF+a in testing. First, the time complexity of evaluating a prediction with FISM (*cf.* Equation (6)) is $O(k\left|\mathcal{R}_u^+\right|)$, where k represents embedding size and $\left|\mathcal{R}_u^+\right|$ denotes the number of historical items interacted by user u. Compared to FISM, the additional time cost of making a prediction with DeepICF is caused by the hidden layers. For the l-th hidden layer, the multiplication between matrices and vectors is the main operation which can be done in $O(d_{l-1}d_l)$, where d_{l-1} represents the size of the l-th hidden layer and $d_0 = k$. The prediction

layer only involves inner product of two vectors, for which the complexity is $O(d_L)$. As such, the overall time complexity for evaluating a DeepICF model is $O(k\left|\mathcal{R}_u^+\right| + \sum_{l=1}^{L} d_{l-1}d_l)$. As reported in [15], the time complexity of NAIS model is $O(k'k\left|\mathcal{R}_u^+\right|)$ where k' denotes the attention factor. For the model of DeepICF+a, the additional time cost comes from the fully connected networks compared to NAIS. Therefore, the overall time cost of evaluating a prediction with DeepICF+a is $O(k'k\left|\mathcal{R}_u^+\right| + \sum_{l=1}^{L} d_{l-1}d_l)$.

3.4 Connections with Other Models

It is worthwhile to point out that FISM can be interpreted as a special case of our proposed DeepICF model. We (i) remove the hidden layers (*i.e.,* set the layer depth $L = 0$) and simply set $\mathbf{e}_L = f_{avg}(\mathcal{V}_{ui})$ in Equation (13), and (ii) project the vector into the prediction layer as,

$$\hat{y}_{ui} = \mathbf{z}^\top(\frac{1}{(|\mathcal{R}_u^+| - 1)^\alpha}(\sum_{j \in \mathcal{R}_u^+ \setminus i} \mathbf{q}_j \odot \mathbf{p}_i)), \tag{16}$$

where $\mathbf{z}$ denotes the weight vector of the prediction layer. Then if we set $\mathbf{z}$ as the all-one vector, DeepICF can exactly recover the FISM model. Obviously, the deep and nonlinear architecture of hidden layers enables DeepICF to investigate the higher-order and nonlinear feature interactions, while the linear modeling limits the capacity of FISM.

Analogously, NAIS is an instance of our proposed DeepICF+a. In particularly, if we (i) set $\mathbf{e}_L$ as the output of attention-based pooling $f_{att}(\mathcal{V}_{ui})$ in Equation (10) and (ii) feed the vector into the prediction layer as,

$$\hat{y}_{ui} = \mathbf{z}^\top \frac{1}{(|\mathcal{R}_u^+| - 1)^\alpha} \sum_{j \in \mathcal{R}_u^+ \setminus i} a(\mathbf{q}_j \odot \mathbf{p}_i) \cdot (\mathbf{q}_j \odot \mathbf{p}_i), \tag{17}$$

then DeepICF+a can recover the NAIS model. Clearly, by taking advantages of nonlinear hidden layers, DeepICF+a not only identifies the importance of feature interactions, but also models the higher-order feature dependencies, which NAIS fails to capture.

4 EXPERIMENTS

In this section, we conduct plenty of experiments on two publicly accessible datasets to answer several questions as follows, which aim at certifying the effectiveness of our proposed methods:

- **RQ1** How do our proposed models (DeepICF and DeepICF+a) perform compared to other state-of-the-art recommender models?
- **RQ2** How do the key hyper-parameter settings impose influence on the performance of our DeepICF models?
- **RQ3** Are deeper layers of hidden units useful for capturing the higher-order and non-linear interactions between items and enhancing the expressiveness of FISM?

Hereinafter, we first describe the settings of experiments followed by answering the aforementioned questions one by one.

4.1 Experimental Settings

Dataset Description. We evaluate the performance of our proposed methods on two real-world datasets. *MovieLens* is a dataset of movie rating which has been leveraged extensively to investigate the performance of CF algorithms. In our experiments, we choose the version including one million ratings where there are 20 ratings per user at least. *Pinterest* is a dataset that is constructed for

Table 1. Statistics of datasets.

Dataset	#Interactions	#Users	#Items	Density
MovieLens	1,000,209	3,706	6,040	4.47%
Pinterest	1,500,809	9,916	55,187	0.27%

content-based image recommendation. The original Pinterest is extremely sparse. We filter the original data similar to MovieLens which keeps each user with at least 20 interactions so as to make it easier to evaluate CF algorithms. These two datasets are publicly accessible on the websites[1] [2]. The detailed characteristics of the two datasets are summarized in Table 1.

Evaluation Protocols. The extensively used *leave-one-out* evaluation protocol [16, 29] is employed here to study the performance of item recommendation. We sort the user-item interactions by the timestamps for each user at first; then we held-out the latest interaction as the testing data for each user and utilized the rest of interactions corresponding to the user for training. Following [16, 20], we sample randomly 99 items (negative instances) which are not interacted by corresponding user for each testing item (positive instance) so as to rank the testing item among such 100 items. As such, we can alleviate the time-consuming problem of ranking all items for each user during evaluation. In terms of evaluation metrics, we adopt *Hit Ratio* at rank k (HR@k) [16] and *Normalized Discounted Cumulative Gain* at rank k (NDCG@k) [1, 5, 12, 14] to evaluate the performance of the ranked list generated by our models. In experimental parts we set $k = 10$ for both metrics. The metric of HR@10 is capable of measuring intuitively if the test item is present at the top-10 ranked list and NDCG@10 illustrates the quality of ranking which assigns higher score to hits at top position ranks. We report the average scores of both evaluation metrics. The higher both metrics are, the better recommendation performance is.

Baselines. To evaluate the efficacy of our proposed models, we also study the performance of the following approaches:

ItemPop. It is non-personalized since it ranks items according to their popularity which is measured by the number of interactions.

ItemKNN [32]. Item-based k-nearest-neighbor (ItemKNN) is the standard item-based CF approach as shown in Equation (1). In the experiments, we attempt to test different numbers of nearest item neighbors, finding that utilizing all neighbors provide best performance.

HOSLIM [8]. This model extends SLIM to capture higher-order item relations. HOSLIM learns two sparse aggregation coefficient matrices for the purpose of capturing the item-item and itemset-item similarities.

Youtube Rec [10]. A deep neural network architecture for recommending YouTube videos. It maps a sequence of video IDs to a sequence of embeddings and these are simply averaged and fed into a feed-forward neural network. The input layer is followed by several layers of fully connected Rectified Linear Units (ReLU).

BPR [29]. This approach optimizes MF model with a pairwise Bayesian Personalized Ranking loss to learn from implicit feedback data. It is often opted for the baseline for item recommendation.

eALS [17]. It is a state-of-the-art MF approach for item recommendation with point-wise regression loss, treating all non-observed interactions as negative instances and weighting them with the corresponding item's popularity.

MLP [16]. This method exploits a multi-layer perceptron instead of the simple inner product to learn the non-linear interactions between users and items from data and tailors a point-wise log

[1] https://grouplens.org/datasets/movielens/1m/
[2] https://sites.google.com/site/xueatalphabeta/academic-projects

loss function to optimize. The experimental results illustrated in later sections are brought by a MLP with three hidden layers.

FISM [19]. This is the state-of-the-art item-based CF method. We experiment with a range from 0 to 1 with a step size of 0.1, discovering setting α as the value of 0 brings about best results.

All the approaches mentioned above cover a various range of recommendation methods: ItemKNN and FISM are on behalf of conventional item-based CF methods to verify the effectiveness of our proposed deep models; BPR and eALS are two competitive user-based methods for implicit feedback; MLP is a recently proposed CF approach based on deep neural networks. In this paper, we primarily pay close attention to single CF models. Hence, we do not make a comparison with NeuMF which is an ensemble model that combines MF with MLP.

Parameter Settings. To avoid overfitting, we tuned the regularization coefficient λ in the range of $[1e^{-7}, 1e^{-6}, \ldots, 1e^{-1}, 1, 10]$ for each learning-based approach. As for the embedding size k, we evaluated the values of $[8, 16, 32, 64]$ in our experiments. For a fair comparison, we trained FISM by optimizing the same objective function of binary cross-entropy loss with the optimizer Adagrad. For our DeepICF models, we initialized them with FISM embeddings which resulted in better performance and faster convergence. And we randomly initialized other model parameters with a Gaussian distribution wherein the value of mean and standard deviation is 0 and 0.01 respectively. The learning rate was searched in $[0.001, 0.05, 0.01]$ and the value of α was experimented in the range of $[0, 0.1, \ldots, 0.9, 1]$. The smooth hyper-paprameter β is consistent with the value when best results are achieved in He's [15] work. Without additional explanation, we leveraged three hidden layers for MLP structure. We implemented our DeepICF models based on Tensorflow[3], which will be released publicly once acceptance.

4.2 Performance Comparison (RQ1)

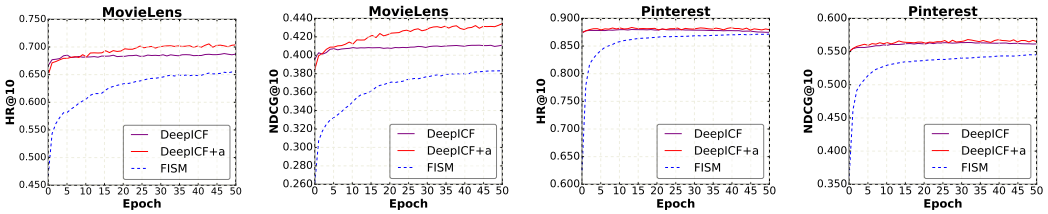
We first make a comparison between our proposed models and other item recommendation approaches. For the purpose of fair comparison, the embedding size is set to 16 for all embedding-based approaches (YouTube Rec, MLP, BPR, eALS, FISM, DeepICF and DeepICF+a). In next subsection, we will alter the embedding size of these embedding-based approaches to observe embedding size performance trends. Table 2 demonstrates the performance of HR@10 and NDCG@10 of all compared methods.

At first, we can see that our DeepICF and DeepICF+a provide the best performance (the highest HR and NDCG scores) on both datasets, significantly outperforming the state-of-the-art item-based method FISM. We attribute such improvements to the effective learning of higher-order item interactions based on deep neural networks and the efficacious introduction of attention mechanism to differentiate the importance of historical items in users' representation. Further more, we conduct one-sample t-tests to justify that all of enhancements are statistically significant with p-Value < 0.05. Secondly, it is obvious to see that the learning-based methods come up with more accurate recommendations than the heuristic-based methods ItemPop and ItemKNN. Especially, HOSLIM captures the linear high-order relation by learning the similarity between itemset and an item, which will limit the performance. What's more, the itemset is generated by the frequent mining algorithm, which requires a support threshold that is non-trivial to tune for different datasets. From Table 2, we can see that our DeepICF and DeepICF+a significantly outperform HOSLIM on both datasets which demonstrate the importance of the non-linear relationship between the higher-order item interactions. FISM significantly exceeds its counterpart ItemKNN with about relative improvements of 6.1% and 18.3% in terms of HR and NDCG on MovieLens. Given the key difference between such two kinds of approaches lies in the way of item similarity estimation, we

[3]https://github.com/AaronHeee/Neural-Attentive-Item-Similarity-Model

Table 2. Performance of HR@10 and NDCG@10 of compared approaches at embedding size 16 and significance test is based on HR@10.

Dataset	MovieLens			Pinterest		
Methods	**HR@10**	**NDCG@10**	***p*-Value**	**HR@10**	**NDCG@10**	***p*-Value**
ItemPop	0.4558	0.2556	9.0e-3	0.2742	0.1410	6.8e-4
ItemKNN	0.6300	0.3341	1.3e-2	0.7565	0.5207	3.9e-4
HOSLIM	0.6851	0.4238	9.6e-6	0.8655	0.5551	1.5e-3
Youtube Rec	0.6874	0.4288	1.2e-3	0.8634	0.5394	6.9e-3
MLP	0.6841	0.4103	1.6e-3	0.8648	0.5385	3.7e-2
BPR	0.6674	0.3907	1.2e-3	0.8628	0.5406	6.4e-4
eALS	0.6689	0.3977	2.8e-3	0.8755	0.5449	5.8e-3
FISM	0.6685	0.3954	5.4e-3	0.8763	0.5529	8.0e-5
DeepICF	**0.6881**	**0.4113**	-	**0.8806**	**0.5631**	-
DeepICF+a	**0.7084**	**0.4380**	-	**0.8835**	**0.5666**	-

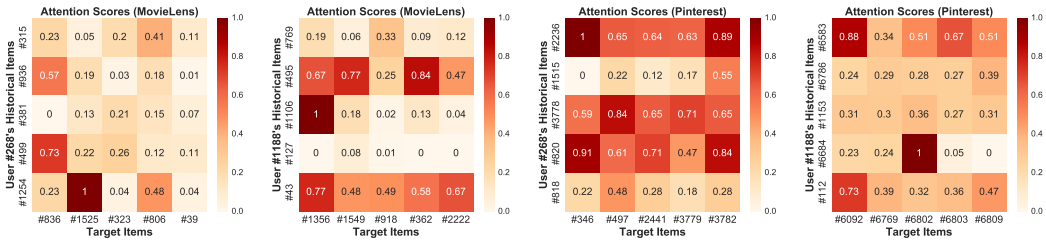


(a) MovieLens – HR@10 (b) MovieLens – NDCG@10 (c) Pinterest – HR@10 (d) Pinterest – NDCG@10

Fig. 5. Testing performance of FISM, DeepICF and DeepICF+a at embedding size 16 in each epoch.

can come to a conclusion that it is extremely important to tailor optimization for recommendation task. Lastly, there is no absolute superior between user-based CF methods (BPR, eALS and MLP) and item-based CF method (FISM). YouTube Rec performs roughly the same as DeepICF on Movielens, weaker than DeepICF+a, but worse on Pinterest. We find the reason is that Pinterest has no time information, while Youtube Rec relies on the chronological order of browsing history. Moreover, we suspect that another reason might be the manner of modeling item relations. Both YouTube Rec and DeepICF utilize the historically interacted items to represent a user, which enriches the input of representation learning. However, our DeepICF utilizes the designed "deep interaction layer" to capture higher-order and nonlinear feature interactions between any two historical items, while YouTube Rec only combines the features of historical items via the meaning pooling operation, which does not explicitly capture feature interactions. In particular, user-based CF methods yield better performance than FISM on the dataset of MovieLens while FISM exceeds the user-based CF methods on Pinterest. We can conclude that it is more predominant for item-based CF model to generate better performance on highly sparse datasets than user-based CF models.

Figure 5 demonstrates the state of DeepICF, DeepICF+a and FISM at embedding size 16 on two datasets in the first 50 epochs. We can obviously find the effectiveness of our proposed models. In particular, the performance of initialized DeepICF and DeepICF+a exceeds significantly FISM in the first epoch. As the training goes on, the experimental results can be even better. Upon convergence, our two DeepICF methods attain relative improvements of 4.8% and 6.6% over FISM



(a) Target User #268 (b) Target User #1188 (c) Target User #268 (d) Target User #1188

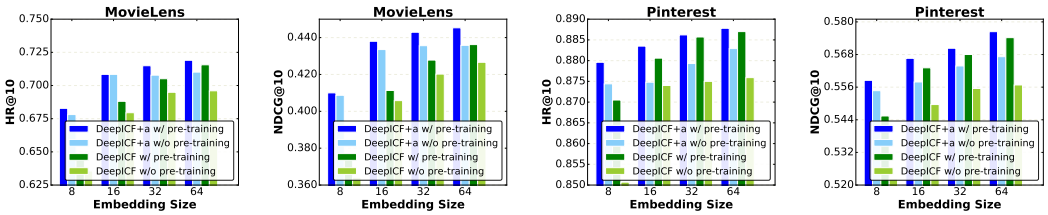
Fig. 6. Visualization of attention weights for each (i, j) pair between target items i and one of historical items of a sampled user j produced by DeepICF+a in MovieLens and Pinterest. Each entry of two heat maps on the left visualizes the attention value a_{ij} of two sampled users from MovieLens on five target items. And another two heat maps on the right for Pinterest are on the same.

Table 3. The prediction scores (after sigmoid) of DeepICF+a and DeepICF. There are two sampled users per datasets and five target items per sampled user. Each user has historically interacted with five items as shown in Figure 6.

MovieLens										
Target Users	**#268**					**#1188**				
Target Items	#836	#1525	#323	#806	#39	#1356	#1549	#918	#362	#2222
$\delta(\hat{y}_{ui})_{\textbf{DeepICF+a}}$	0.51	0.52	0.56	0.80	0.70	0.64	0.52	0.70	0.62	0.67
$\delta(\hat{y}_{ui})_{\textbf{DeepICF}}$	0.19	0.20	0.37	0.61	0.53	0.31	0.37	0.48	0.42	0.57
Pinterest										
Target Users	**#268**					**#1188**				
Target Items	#346	#497	#2441	#3779	#3782	#6092	#6769	#6802	#6803	#6809
$\delta(\hat{y}_{ui})_{\textbf{DeepICF+a}}$	0.32	0.66	0.41	0.34	0.72	0.49	0.52	0.17	0.63	0.69
$\delta(\hat{y}_{ui})_{\textbf{DeepICF}}$	0.18	0.62	0.22	0.30	0.68	0.39	0.50	0.10	0.39	0.59

in terms of NDCG on the datasets of MovieLens and Pinterest, respectively. Such promising results reveal the reason why both DeepICF models are capable of providing much better recommendation performance than FISM, proving the key arguments of our work that the higher-order relations between items can be better modeled based on deep neural networks and different interacted items of a user should be weighted differently in contributing to the preference of corresponding user.

4.2.1 ***Explainability.*** To demonstrate the explainability of our enhanced model DeepICF+a which introduces attention weights to differentiate contributions of differently historical items corresponding to a particular user for the final prediction, we separately sampled two users from two datasets of MovieLens and Pinterest which are positive examples and the corresponding scores should be predicted larger. Meanwhile, we select five historically interacted items for each user. Figure 6 visualizes the attention weight learned by DeepICF+a model in which a row denotes a historically interacted item of a sampled user and a column denotes a target item. The two heat maps on the left in Figure 6 presents the attention scores over five selected items of two users sampled from MovieLens. Another two heat maps on the right for the datasets of Pinterest are on the same.

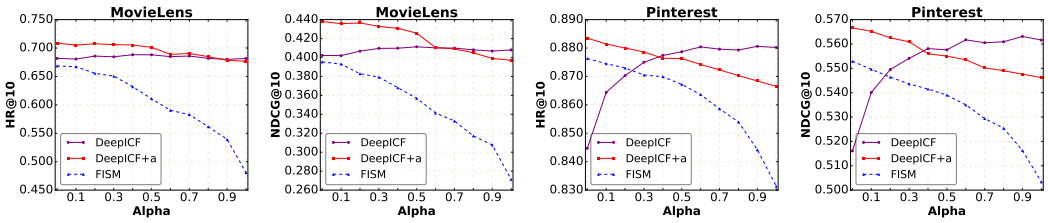


(a) MovieLens – HR@10 (b) MovieLens – NDCG@10 (c) Pinterest – HR@10 (d) Pinterest – NDCG@10

Fig. 7. Performance of both DeepICF methods with (w/) and without (w/o) FISM item embeddings pre-training.

Taking the user #1188 sampled from MovieLens and the corresponding target item #1549 for an example. We can clearly see that DeepICF weights all the historical items (Item #43, #127, #1106, #495 and #769 in this case) of user #1188 equally while DeepICF+a assigns different weights to the five historical items. In more concrete terms, DeepICF+a assigns higher attention scores over item #495 and #43 but relatively lower attention scores for the rest of three items. As shown in Table 3, the prediction score (after sigmoid) of DeepICF+a on item #1549 in MovieLens is 0.52, which is higher than the score 0.37 predicted by DeepICF on the corresponding target item #1549. To better explain the rationality, we further give an insight into these movies from MovieLens. We found that these three movies of #1549, #495 and #43 are all about romance drama movies while movie #769 and #1106 are documentary. Although movie #127 is also a drama movie, its attention weight is quite low. We deem the reason is that the main content of movie #127 is about social enslavement not romance. It is quite reasonable when predicting the preference of a user on a target item, all of her historical items whose categories are similar to the target item, should impose more impact on the final prediction than the irrelevant ones. DeepICF+a successfully predicts the score higher which is expected strongly. Such learning results verify the usefulness of attention mechanism introduction into our proposed DeepICF model.

4.2.2 ***Utility of Pre-training.*** It is significant for deep learning models in terms of parameter initialization which can impose impact on their convergence and final performance [11], [16]. Owing to the non-convexity of our DeepICF methods' objective functions, we make a comparison between two DeepICF methods with and without pre-training to certify the effectiveness of utility of item embeddings pre-training (*i.e.,* leveraging the item embeddings learned by FISM to initialize the corresponding item embeddings of DeepICF models). As for the version of both DeepICF models without pre-training, we leveraged the Adagrad optimizer to learn the corresponding models with item embeddings initializing randomly. While for DeepICF and DeepICF+a with pre-training, we first run FISM till convergence and then exploit its item embeddings to initialize the corresponding item embeddings of DeepICF and DeepICF+a. As we can see from the Figure 7, both DeepICF models initialized by FISM embeddings provide much better recommendation performance than the ones without pre-training. For instance, the relative enhancements of the DeepICF with pre-training over the one trained from random initialization at embedding size 16 are 1.3% and 0.8% in terms of HR on the datasets of MovieLens and Pinterest, respectively. Furthermore, DeepICF models with FISM embeddings initialized are able to converge much faster than the ones without pre-training. This experimental results prove the efficacy of pre-training for two DeepICF models.



(a) MovieLens – HR@10 (b) MovieLens – NDCG@10 (c) Pinterest – HR@10 (d) Pinterest – NDCG@10

Fig. 8. Performance of both DeepICF models *w.r.t* the normalization constant α (embedding size = 16).

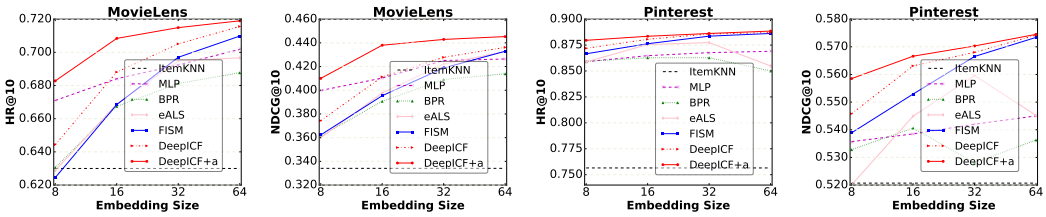
4.3 Sensitivity to Hyper-parameter (RQ2)

In this study, we investigate the impact of different values of normalization hyper parameter α and different negative sampling ratios on the performance of our both DeepICF models. In addition, as embedding-based models, the embedding size is a critical hyper-parameter as well. In this subsection, we are also ready to compare the influence of different embedding sizes on the performance trends.

4.3.1 ***Effect of Normalization Coefficient*** α. Figure 8 demonstrates the performance of both DeepICF methods with regard to the normalization hyper-parameter α. Keeping the rest parameters constant, we did a full parameter study for different values of α. From the experimental results, the performance of HR and NDCG corresponding to FISM decreases gradually with the increase of α with step size of 0.1. For the performance of DeepICF, we can see that the best result on MovieLens appears in the range 0.4 to 0.5 and exceeds FISM no matter what the setting of α. While on Pinterest, the best result is obtained when the value of α is in the range 0.5 to 1 and outperforms FISM when the value of α is 0.2. For the enhanced version, DeepICF+a achieves best performance on both datasets when α is set to the value of 0 and outperforms DeepICF. We contribute such improvements to the introduction of attention mechanism into DeepICF.

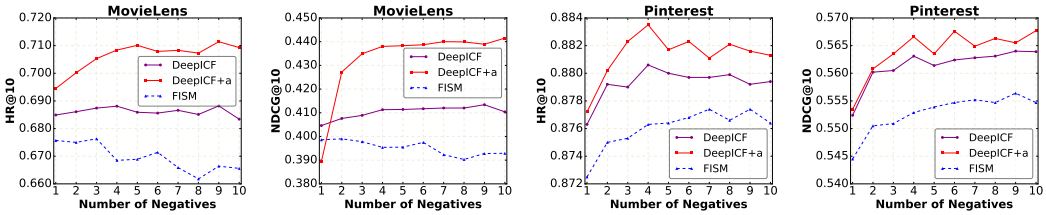
4.3.2 ***Effect of Item Embedding Size***. Figure 9 shows the performance of HR and NDCG with respect to the embedding size. As we can see that the tendencies of recommendation performance at embedding size 8, 32 and 64 are similar to the one at embedding size 16 in general. Our proposed DeepICF approach outperforms all the other methods under most circumstances except for embedding size 8 where MLP achieves better performance than DeepICF on MovieLens. We argue that on the relative dense dataset of MovieLens (compared to Pinterest), user-based non-linear methods (MLP in this case for instance) have the ability to express stronger representation at small embedding size. Our enhanced model DeepICF+a offers the best performance and covers the shortage of DeepICF for prediction at small embedding size in dense dataset.

4.3.3 ***Effect of Negative Instance Sampling***. To illustrate the impact for DeepICF and DeepICF+a with regarding to negative instance sampling, we demonstrate the experimental results of both DeepICF methods with different ratios of negative sampling in Figure 10. We can find that the performances of two DeepICF models are better than FISM when just sampling one negative instance per positive instance and sampling more negative instances seems more helpful to improve performances for DeepICF and the enhanced model DeepICF+a. For these two datasets, the optimal number of negative samples per positive instance for DeepICF models is around 4 which is similar to the experimental results by He's work [16].



(a) MovieLens – HR@10 (b) MovieLens – NDCG@10 (c) Pinterest – HR@10 (d) Pinterest – NDCG@10

Fig. 9. Performance of HR@10 and NDCG@10 *w.r.t* the embedding size on the two datasets.



(a) MovieLens – HR@10 (b) MovieLens – NDCG@10 (c) Pinterest – HR@10 (d) Pinterest – NDCG@10

Fig. 10. Performance of both DeepICF models *w.r.t* the number of negative instances per positive instance (embedding size = 16).

4.4 Depth of Hidden Layer in Network (RQ3)

Our proposed models capture the higher-level and non-linear relations between items by virtue of neural networks with deep hidden layers. The hidden layers of our models play a vital part in learning higher-order item interactions in the way of non-linearity. As there is relatively little work on learning the complex interaction function between items based on deep neural networks, it is curious to see whether leveraging a deep architecture is propitious to bring about encouraging performance in modeling higher-order relations in a non-linear way for quality prediction or not. In the final part of experiments, we further investigate DeepICF with different number of hidden layers, omitting the DeepICF+a model here owing to space limitation.

The experimental results are provided in the Table 4. The DeepICF-4 for instance, refers to the DeepICF model with four hidden layers and other notations are in the same. As we can see that to some extent it is beneficial for DeepICF to stack more non-linear hidden layers so as to better capture the higher-order relations between items and then generate promising performance. This result is highly encouraging, indicating the effectiveness of using deep architecture for complex item relations learning. Such advance is credited to the higher-order and non-linear item relations brought by stacking more non-linear hidden layers. In order to certify this, we further attempted to stack linear layers, replacing ReLU with an identity function as the activation function of hidden layers. The corresponding experimental performances are much worse than utilizing ReLU function. This provides evidence to the necessity of learning higher-order interactions between items with non-linear functions. Owing to space limitation, we omit the results of using identity function as activation function.

Table 4. HR@10 and NDCG@10 of DeepICF with different hidden layers (embedding size = 16). The best result is highlighted as bold font.

Embedding Size	DeepICF-1		DeepICF-2		DeepICF-3		DeepICF-4	
	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG
MovieLens								
8	0.6424	0.3707	0.6444	0.3740	0.6444	0.3741	**0.6444**	**0.3743**
16	0.6833	0.4081	0.6854	0.4086	0.6881	**0.4113**	**0.6884**	0.4107
32	0.7022	0.4236	**0.7051**	0.4278	0.7048	0.4276	0.7050	**0.4320**
64	0.7116	0.4327	0.7131	0.4386	**0.7156**	0.4362	0.7124	**0.4388**
Pinterest								
8	0.8679	0.5379	0.8692	0.5415	0.8705	0.5454	**0.8719**	**0.5458**
16	0.8804	0.5547	0.8792	0.5607	0.8806	**0.5631**	**0.8810**	0.5608
32	0.8844	0.5669	0.8852	0.5654	**0.8857**	0.5680	0.8844	**0.5692**
64	0.8858	0.5708	0.8865	0.5691	0.8865	0.5720	**0.8870**	**0.5742**

5 RELATED WORK

The core to a personalized recommender system lies in collaborative filtering, namely modeling the preference of users towards items according to the their historical interactions. In this section, we briefly review the related literature about collaborative filtering from the following three aspects.

5.1 User-based Collaborative Filtering Models

UCF has been extensively investigated in bost academia and industry. The UCF task with explicit feedback (*e.g.,* user ratings), which directly reflects the preference of users on items, is usually formulated as a rating prediction problem [20, 32]. The target is to minimize the overall errors between the known ratings and the corresponding prediction scores. Among various UCF approaches, matrix factorization has been the frequently praised model due to its simplicity and effectiveness. Biased MF is proposed to further enhance the performance of traditional MF in the problem of rating prediction. Researchers in [23, 24, 26, 35, 37, 40] introduced extra information like review texts and social relations into MF so as to address the rating sparsity issue. Among numerous MF-based approaches, SVD++ has been proven to be the best single model in terms of fitting user ratings. SVD++ firstly factorizes the user-item rating matrix with implicit feedback [20] and is followed by lots of techniques for recommender systems [13, 29, 39].

Since user-item interactions on many recommender systems are based on implicit feedback (*e.g.,* view, click) rather explicit ratings, many approaches are proposed on the basis of implicit feedback [3, 13, 16, 17, 19, 28]. UCF with implicit feedback is usually treated as top-N recommendation task [22], which offers a short ranked list of items to the potential users. Technically, the main difference between the tasks of rating prediction and top-N recommendation lies in the way of model optimization. In particular, the former usually constructs a regression loss function only on the known ratings to optimize, yet the latter needs to take the remaining data (a mixture of real negative feedback and missing data) into consideration which are always ignored by the models for explicit feedback. Recently, researchers in [17] presented a well-designed MF-based method which applied a popularity-aware weighting strategy to model these remaining data, achieving state-of-the-art performance for top-N recommendation.

5.2 Item-based Collaborative Filtering Models

ICF has been used for the construction of industrial applications on online recommendation due to the excellent efficacy it performs. The core of item-based CF is the estimation of item-item similarities. Early heuristic-based models (*e.g.,* ItemKNN [32]) simply utilize the statistical measures like cosine similarity and Pearson correlation coefficient to estimate similarities between items. However, such methods requires extensive manual tuning to measure similarity well and hardly generalize to other datasets. To solve these issues, there appears some machine learning-based approaches attempting to construct objective function to automatically learn the item-item similarity matrix. Among these item-to-item learning-based methods, SLIM [27] and FISM [19] are two representative models. Specifically, SLIM learns the item similarity matrix by building a regression-based objective function to optimize. However, it suffers from high training cost and fails to capture the transitive relationships between items. What's more, the work in [8] extends item similarities to high orders and captures higher-order item relations by integrating itemsets to SLIM. As for FISM, it factorizes the similarity between two items as the inner product of their low-dimensional vectors. While achieving the state-of-the-art performance, FISM has two inherent limitations. One is that FISM only model the second-order item-item similarity relations via the inner product but ignores the complex higher-order relationships between items. Another is that FISM assumes all historically interacted items of a user profile contribute equally for modeling user's preference on the target item.

The pioneering work based on neural network for ICF is the collaborative denoising autoencoder (CADE) presented by Wu *et al.*[42]. It is worth mentioning that CADE can recover SVD++ when replacing the activation function of hidden layers with a identity function. CADE is a neural modeling method for ICF, however it still leverages the linear inner product to model the user-item interactions, limiting its expressiveness and capacity to capture the nonlinear relations.

5.3 Deep Collaborative Filtering Models

More recent evidence has suggested that integrating deep learning with recommender systems can significantly boost the performance [4, 45]. Salakhutdinov *et al.*[31] firstly proposed to exploit two-layer Restricted Boltzmann Machines to model the ratings of users on items. Autoencoders and the denoising autoencoders have already applied for recommendation based on explicit feedback [21, 33]. In addition, there are also some recent works [7, 38, 44] attempting to leverage deep neural networks for feature extraction from the side information of images and music and then integrating these features with MF models. It is obvious that these advances still belong to the family of shallow and linear models. To address the core technology of CF based on deep neural networks, He *et al.*[16] initiate a general CF framework named NCF, which uses feed-forward neural networks to model user-item interactions, instead of linear inner product. Based on the NCF framework, Ting *et al.*[2] integrate the localized information (*i.e.,* user and item neighborhood information) to enrich the representations. Govington *et al.*[10] proposed a deep neural network structure for industrial recommendation system. It considers recommendation as extreme multiclass classfication where the prediction problem becomes classifying a user into a video in a specific context. As a practical recommendation system, it can model newly upload content by feeding the age of example into neural network. More recently, He *et al.*[14] present a neural network view for factorization machine, named NFM, which captures the nonlienar and higher-order feature interactions by the proposed bilinear layer.

Inspired by NCF [16] and NFM [14], our DeepICF and DeepICF+a take advantages of neural networks to automatically learn the complex interaction function between items. Specifically, the deep component of DeepICF models applies a standard structure of MLP above the item embedding

vectors, similar to that of NCF [16]. It will be our future work with regarding to the choice of DNNs' architecture. In this work, we demonstrate that DNNs can be a promising choice for item-based CF about modeling the higher-order and non-linear interactions between items.

6 CONCLUSIONS AND FUTURE WORK

In this work, we present a new item-based CF solution based on deep neural networks named DeepICF for top-N item recommendation. Our key argument is that the potential structure of real-world data tends to be greatly non-linear and is incapable of being approximated accurately by linear models such as FISM. The proposed model can not only conquer the inherent limitations of FISM successfully, but also effectively learn the higher-order relations among items from data in a non-linear way of neural networks. We conduct a comprehensive set of experiments on two real-world datasets and the corresponding experimental results demonstrates that DeepICF outperforms other state-of-the-art item-based approaches for top-N item recommendation task.

In future, we plan to improve DeepICF in three directions. First of all, this work focuses on modeling item relations based solely on implicit similarity, however there indeed exist many heterogeneous item relations, which can be characterized based on attributes (*e.g.,* category, location) or other content (*e.g.,* time-stamp, co-occurrence). Hence we attempt to identify the relational knowledge between items and improve DeepICF with such item relations. Second, although DeepICF can offer explanations behind a recommendation, like "item A is recommended since you have experienced similar item B", such similarity-based evidence may be too coarse-grained to increase users' trust. We would like to use side information or relational knowledge of users and items to exhibit feature-based reasons. Third, we will investigate the sequential recommendation by modeling the evolution of user preferences towards items via reinforcement learning or tracking user tastes on different attributes of items via memory network.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (No. 61772170, 61472115), the National Key Research and Development Program of China (No. 2017YFB0803301) and the Fundamental Research Funds for the Central Universities (No. JZ2017YYPY0234). This work is also supported by the National Research Foundation, Prime Ministers Office, Singapore under its IRC@Singapore Funding Initiative. The authors would like to thank the anonymous reviewers for their reviewing efforts and valuable comments.

REFERENCES

[1] Avi Arampatzis and Georgios Kalamatianos. 2018. Suggesting Points-of-Interest via Content-Based, Collaborative, and Hybrid Fusion Methods in Mobile Devices. *ACM Trans. Inf. Syst.* 36, 3 (2018), 23:1–23:28.

[2] Ting Bai, Ji-Rong Wen, Jun Zhang, and Wayne Xin Zhao. 2017. A Neural Collaborative Filtering Model with Interaction-based Neighborhood. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 1979–1982.

[3] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A generic coordinate descent framework for learning from implicit feedback. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1341–1350.

[4] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H. Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*. 46–54.

[5] Da Cao, Xiangnan He, Liqiang Nie, Xiaochi Wei, Xia Hu, Shunxiang Wu, and Tat-Seng Chua. 2017. Cross-Platform App Recommendation by Jointly Modeling Ratings and Texts. *ACM Trans. Inf. Syst.* 35, 4 (2017), 37:1–37:27.

[6] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention. In *Proceedings of the 40th*

International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017. 335–344.

[7] Xu Chen, Yongfeng Zhang, Qingyao Ai, Hongteng Xu, Junchi Yan, and Zheng Qin. 2017. Personalized Key Frame Recommendation. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. 315–324.

[8] Evangelia Christakopoulou and George Karypis. 2014. Hoslim: Higher-order sparse linear method for top-n recommender systems. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 38–49.

[9] Evangelia Christakopoulou and George Karypis. 2016. Local Item-Item Models For Top-N Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 67–74.

[10] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.

[11] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research* 11, Feb (2010), 625–660.

[12] Jiangning He and Hongyan Liu. 2017. Mining Exploratory Behavior to Improve Mobile App Recommendations. *ACM Trans. Inf. Syst.* 35, 4 (2017), 32:1–32:37.

[13] Ruining He and Julian McAuley. 2016. VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. 144–150.

[14] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 355–364.

[15] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. NAIS: Neural Attentive Item Similarity Model for Recommendation. *IEEE Transactions on Knowledge and data Engineering* (2018).

[16] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 173–182.

[17] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 549–558.

[18] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2, 5 (1989), 359–366.

[19] Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 659–667.

[20] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 426–434.

[21] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 811–820.

[22] Xin Li, Mingming Jiang, Huiting Hong, and Lejian Liao. 2017. A Time-Aware Personalized Point-of-Interest Recommendation via High-Order Tensor Factorization. *ACM Trans. Inf. Syst.* 35, 4 (2017), 31:1–31:23.

[23] Defu Lian, Kai Zheng, Yong Ge, Longbing Cao, Enhong Chen, and Xing Xie. 2018. GeoMF++: Scalable Location Recommendation via Joint Geographical Modeling and Matrix Factorization. *ACM Trans. Inf. Syst.* 36, 3 (2018), 33:1–33:29.

[24] Yi Liao, Wai Lam, Lidong Bing, and Xin Shen. 2018. Joint Modeling of Participant Influence and Latent Topics for Recommendation in Event-based Social Networks. *ACM Trans. Inf. Syst.* 36, 3 (2018), 29:1–29:31.

[25] David C Liu, Stephanie Rogers, Raymond Shiau, Dmitry Kislyuk, Kevin C Ma, Zhigang Zhong, Jenny Liu, and Yushi Jing. 2017. Related pins at pinterest: The evolution of a real-world recommender system. (2017), 583–592.

[26] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 165–172.

[27] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*. IEEE, 497–506.

[28] Mirko Polato and Fabio Aiolli. 2018. Boolean kernels for collaborative filtering in top-N item recommendation. *Neurocomputing* 286 (2018), 214–225.

[29] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[30] Steffen Rendle and Lars Schmidt-Thieme. 2008. Online-updating Regularized Kernel Matrix Factorization Models for Large-scale Recommender Systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems*. ACM.

[31] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*. ACM, 791–798.

[32] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. ACM, 285–295.

[33] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 111–112.

[34] Ying Shan, T. Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. 2016. Deep Crossing: Web-Scale Modeling Without Manually Crafted Combinatorial Features. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 255–262.

[35] Lei Shi, Wayne Xin Zhao, and Yi-Dong Shen. 2017. Local Representative-Based Matrix Factorization for Cold-Start Recommendation. *ACM Trans. Inf. Syst.* 36, 2 (2017), 22:1–22:28.

[36] Brent Smith and Greg Linden. 2017. Two decades of recommender systems at Amazon. com. *IEEE Internet Computing* 21, 3 (2017), 12–18.

[37] Yu Sun, Nicholas Jing Yuan, Xing Xie, Kieran McDonald, and Rui Zhang. 2017. Collaborative Intent Prediction with Real-Time Contextual Data. *ACM Trans. Inf. Syst.* 35, 4 (2017), 30:1–30:33.

[38] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Advances in neural information processing systems*. 2643–2651.

[39] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2018. TEM: Tree-enhanced Embedding Model for Explainable Recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*. 1543–1552.

[40] Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2017. Item silk road: Recommending items from information domains to social users. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 185–194.

[41] Zihan Wang, Ziheng Jiang, Zhaochun Ren, Jiliang Tang, and Dawei Yin. 2018. A Path-constrained Framework for Discriminating Substitutable and Complementary Products in E-commerce. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 619–627.

[42] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 153–162.

[43] Hong-Jian Xue, Xin-Yu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep matrix factorization models for recommender systems. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. 3119–3125.

[44] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 353–362.

[45] Shuai Zhang, Lina Yao, and Aixin Sun. 2017. Deep Learning based Recommender System: A Survey and New Perspectives. *CoRR* abs/1707.07435 (2017).

[46] Yongfeng Zhang and Xu Chen. 2018. Explainable Recommendation: A Survey and New Perspectives. *CoRR* abs/1804.11192 (2018).