IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 25, 2020, accepted November 8, 2020, date of publication November 19, 2020,
date of current version December 9, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3039380

Item-Based Collaborative Memory Networks for Recommendation

DEWEN SENG, GUANGSEN CHEN, AND QIYAN ZHANG
School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China
Key Laboratory of Complex Systems Modeling and Simulation, Ministry of Education, Hangzhou 310018, China

Corresponding author: Dewen Seng (sengdw@hdu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61903109, in part by the Natural Science Foundation of Zhejiang Province under Grant LY20F020013, and in part by the Fundamental Public Welfare Research Program of Zhejiang Province under Grant LGF19F020015 and Grant LGG21F020006.

ABSTRACT Item-based Collaborative Filtering (ICF or Item-based CF) has been widely applied for recommender systems in industrial scenarios, owing to its efficiency in user preference modeling and flexibility in online personalization. It captures a user's preference from his or her historically interacted items, recommending new items that are the most similar to the user's preference. Recently, several works propose advanced neural network architectures to learn item similarities from rating data, by modeling items as latent vectors and learning model parameters by optimizing a recommendation-aware objective function. While much literature attempts to use classical neural networks such as multi-layer perception (MLP), to learn item similarities, there has been relatively less work employing memory networks for ICF, which can more accurately record the detailed information about entities than classical neural networks. Therefore, in this paper, we propose a powerful Item-based Collaborative Memory Network (ICMN) for ICF, which bases on the architecture of Memory Networks. Besides, a neural attention mechanism is adopted to focus on the most important historically interacted items. The core of our ICMN is the cooperation of external and internal memory and the contribution of the neural attention mechanism. Compare to the state-of-the-art ICF methods, our ICMN possesses the merit of powerful representation capability. Extensive experiments on two datasets demonstrate the effectiveness of ICMN. To the best of our knowledge, this is the first attempt that applies memory networks for ICF.

INDEX TERMS End-to-end memory networks, item-based collaborative filtering, attention mechanism.

I. INTRODUCTION

In the era of information explosion, the recommender system is an important tool to deal with the problems caused by information overload. It can automatically search and filter the products we need among the massive number of products. On the one hand, it protects users from suffering from being at a loss for the enormous number of selections. On the other hand, it increases the traffic and profits of online services rapidly.

Recently, collaborative filtering (CF), a technique that predicts personalized preference of users from historical user-item interaction only, plays an essential role in modern recommender systems especially in the phase of candidate generation [18], [26], [37]. Popularized by the Netflix Prize, Matrix Factorization (MF) has become the most popular approach to model-based recommendation in academia and has been deeply studied in the literature [19], [47]. Although MF methods have shown superior accuracy over neighbor-based methods in terms of rating prediction, they have seldom been used in industrial applications due to its personalization scheme [17].

Item-based collaborative filtering, as a well-known kind of nearest-neighborhood based method, modeling users' preference by representing a user with his historically interacted items, predicts the interaction between the user and new items based on the item-item similarity. Comparing to MF which represents a user by a simple ID, ICF offers more detailed information about a user's profile, which also provides more interpretable prediction in various recommendation scenarios. It is more suitable for real-time application than MF as the major process that computes item similarity can be processed

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

offline and the only thing that the recommendation module needs to do is to perform a series of lookups on similar items that user historically interacted. Typically, ICF has been widely used in industrial applications [7], [13], [26], [35].

Primitive ICF methods use statistical measures such as Pearson coefficient and cosine similarity to quantify item similarities. Such methods typically underperform machine learning-based methods in top-K rank tasks [32]. Therefore, to overcome the drawback, the authors of SLIM (Sparse Linear Method) [28] combined the merits of model-based method and ICF method by optimizing a recommendation-aware objective function. Although SLIM achieved better accuracy, learning the whole item-item similarity matrix directly is quite time-consuming and the item similarity can not be transmitted between items. In FISM (Factored Item Similarity Model) [20], each item is represented by a latent vector. To estimate the similarity value between a pair of items, we only need to compute the inner product of the two latent vectors. Therefore, the similarity between items can be well transmitted. NAIS (Neural Attentive Item Similarity Model for Recommendation) [17] addressed the inefficiency issue of FISM that considering the importance of items uniformly by using attention network to discriminate which interacted items are more influential. Deep-ICF (Deep Item-based Collaborative Filtering for Top-N Recommendation) [46] argued previous works on ICF only model the second-order relations between items and higher-order relations are discarded. Deep-ICF also illustrates the difference between high-order and second-order relations between pairs of items and come up with an idea that capturing the high-order relations with deep networks. However, these models lack the capacity of memorizing detailed information about historically interacted items to model the complex preference of users, since shopping online becomes convenient with the development of e-commerce and some users may purchase a lot of items. In other words, the traditional neural networks maybe not competent with such tasks as they lose much detailed information when they are modeling user's preference by a massive number of historical items, making it infeasible for recommenders to practically tackle the myriads of changes of user's interests.

In recent years, a surge of interest in using deep learning to recommender systems has emerged. Literature [8], [29], [39] applied DNNs to model auxiliary information. [31] addressed collaborative filtering by applying Restricted Boltzmann. Autoencoders are also popular choices for recommender systems [23], [33], [40], [45], [49]. These deep learning architectures achieved effective performance and demonstrated the advantages of nonlinear transformations over traditional linear models [48]. However, there is little work on employing memory networks for recommender systems compared with the huge number of literature applying other deep learning architectures for recommenders.

In this work, we propose an approach to portray the complicated personalized preference of users through detailed information about numerous historical items. Hence, a strong memory of the model for items is indispensable. Although previous neural networks such as Recurrent Neural Networks (RNN) [44] possess the capability of memory for an entity, the literature [36] argues that memory components of these networks are too small to remember details of a complex entity, and are not compartmentalized enough to accurately remember facts from the past (knowledge is compressed into dense vectors) [43]. Therefore, we develop a model that capitalizes on the advantages of Memory Networks and neural attention mechanisms for Item-based CF with implicit feedback. Complicated preference of users will be modeled with the cooperation of external and internal memory; high-order complex item-item relations will be captured based on multi-layer and non-linear architecture, and the neural attention mechanism places higher weights on specific subsets of historically interacted items which are the most similar to target items. Finally, the interactions between the user and target items are predicted by an output module in memory networks.

Our Item-based Collaborative Filtering (ICMN) model inherits the advantage of FISM in terms of high efficiency in online services, while is more expressive and accurate than FISM due to the complex non-linear transformation architecture.

Our primary contributions are as follows:

1) We utilize Memory Networks for Item-based CF to capture the high-order item-item relation and encode detailed information about items into the memory module, combing with the attention mechanism to predict the user-item interaction.
2) We introduce a small trick submitted in the NAIS [17] called Smooth to address the problem that standard attention mechanism can not learn from users historically interacted items as the large variance on the lengths of user histories.
3) We employ Layer Normalization [1] which seldomly appears in Item-based CF literature to address the vanishing gradient problem.
4) We conduct a series of experiments on two popular real-world datasets to demonstrate the effectiveness.

II. PRELIMINARIES

In this part, we first brief the traditional item-based CF, then recapitulate the SLIM which is the first learning-based ICF. After that, we introduce FISM and NAIS. Both of them are state-of-the-art approaches to ICF. Finally, we analyze Deep-ICF [46] which extends the NAIS.

A. ITEM-BASED COLLABORATIVE FILTERING

Item-based CF [32] calculates the similarity between historically interacted items and the target item by mathematical statistics methods and uses the user's satisfaction for similar items to predict the user's rating score for target items. Formally, the predictive model of item-based CF is:

$$\hat{y}_{ui} = \sum_{j \in R_u^+} r_{uj} S_{ij} \tag{1}$$

where S_{ij} is the similarity score between interacted item j and target item i. r_{uj} is the preference of user u for item j, which can be a real-valued rating score (explicit feedback) and a binary value 1 or 0 (implicit feedback). R_u^+ is the set of items that the user has interacted with.

B. LEARNING-BASED METHOD FOR ITEM-BASED CF

The authors of SLIM [28] came up with a learning-based method for Item-based CF, which learns item-item similarities by building an objective function then optimizing it. The primary idea is to minimize the loss between the ground-truth prediction and the reconstructed one out of the model. Formally, the objective function is as follows:

$$\begin{aligned} L &= \frac{1}{2}||\boldsymbol{A}\boldsymbol{W} - \boldsymbol{A}||_F^2 + \frac{\beta}{2}||\boldsymbol{W}||_F^2 + \lambda||\boldsymbol{W}||_1 \\ \textit{subject to } \boldsymbol{W} &\geq 0, \quad \textit{diag}(\boldsymbol{W}) = 0, \end{aligned} \tag{2}$$

where $\boldsymbol{W}$ is a matrix about the item-item similarity, $\boldsymbol{A}$ is the original rating score matrix. β is the L_2 regularization coefficient which prevents the model from overfitting. And λ is a L_1 regularization coefficient which enforces sparsity on matrix $\boldsymbol{W}$, since there are only a few items that are particularly similar to an item in the real world dataset. Besides, the non-negativity constraint on each value of $\boldsymbol{W}$ is to ensure it is a meaningful similarity metric. The zero constraint on the diagonal values of $\boldsymbol{W}$ is to avoid the impact of the target item itself when the model performs the prediction.

Although recommending accuracy is improved, the inherent drawback of SLIM is obvious. First, refreshing the similarity matrix will cost $O(I^2)$ of time complexity. Second, it can only learn similarities for two items that have been co-rated by the same person before. Therefore, SLIM fails to capture transitive relations between items. To repair the drawback, FISM [20] projects all the items into the same vector dimension, representing an item with a low-dimensional embedding vector. Based on this, the similarity score can be easily measured, which is parameterized as the inner product between the embedding vector of item i and item j. Formally, the predictive model of FISM is:

$$\hat{y}_{ui} = \boldsymbol{p}_i^T \left(\frac{1}{|R_u^+|^\alpha} \sum_{j \in R_u^+ \setminus \{i\}} \boldsymbol{q}_j\right) \tag{3}$$

where $\boldsymbol{p}_i^T$ and $\boldsymbol{q}_j$ represent the embedding vector for item i and j, respectively. The notation $\setminus \{i\}$ means constraint of diag(W) $= 0$, and α is a hyper-parameter which controls the normalization effect. From the perspective of Matrix Factorization, the formulation in the bracket can be seen as an embedding vector of user u. Therefore, to some extend, FISM is quite similar to MF but provides more information about users for the model to capture the preference of users. While FISM achieves advanced performance among item-based CF methods and inherits the advantage of Item-based CF which is suitable for the online recommendation, its equal treatments on all historical items of a user limit its representation ability.

C. ATTENTION-AWARE METHOD

NAIS [17] employs a neural attention network to learn the varying weights of item-item relations on the foundation of FISM, which is formulated as:

$$\hat{y}_{ui} = \sum_{j \in R_u^+ \setminus \{i\}} a_{ij} \boldsymbol{p}_i^T \boldsymbol{q}_j \tag{4}$$

where a_{ij} represents the attentive weight of similarity, which means the importance of interacted item j when the model predicts the interaction between user u and item i. Here, the operation of a_{ij} is different from the standard solution of attention, which is specifically formulated as:

$$\begin{cases} a_{ij} = softmax'(f(\boldsymbol{p}_i, \boldsymbol{q}_j)) \\ f_{concat(\boldsymbol{p}_i, \boldsymbol{q}_j)} = \boldsymbol{h}^T ReLU(\boldsymbol{W}[\boldsymbol{p}_i, \boldsymbol{q}_j] + \boldsymbol{b}) \\ f_{prod(\boldsymbol{p}_i, \boldsymbol{q}_j)} = \boldsymbol{h}^T ReLU(\boldsymbol{W}(\boldsymbol{p}_i \odot \boldsymbol{q}_j) + \boldsymbol{b}) \end{cases} \tag{5}$$

where $softmax'$ is a variant of the standard $softmax$ function that takes the normalization on user history length into account, which can be formulated as:

$$softmax'(f(\boldsymbol{p}_i, \boldsymbol{q}_j)) = \frac{exp(f(\boldsymbol{p}_i, \boldsymbol{q}_j))}{[\sum_{j \in R_u^+ \setminus \{i\}} exp(f(\boldsymbol{p}_i, \boldsymbol{q}_j))]^\beta} \tag{6}$$

where β is the smoothing exponent, a hyperparameter ranging from 0 to 1, which is used to restrict the problem that large variance of user historically interacted items will restrict the learning ability of the attention layer. The basic theory is: $softmax$ function performs L1 normalization on attention weights, which may overly punish the weights of active users with a long history. And smoothing the denominator of $softmax$ can reduce the punishment on attention weights of active users, meanwhile decrease the variance of attention weights.

Besides, the original normalization term $|R_u^+|^\alpha$ in FISM is aborted into the attention weight a_{ij} without losing the representation ability in NAIS. $\boldsymbol{W}$ and $\boldsymbol{b}$ are the weight matrix and bias vector, respectively. $\boldsymbol{h}$ is a weight vector that projects the hidden layer into the scalar output.

D. DEEP-BASED METHOD

Deep-ICF [46] replaces the dot product operation by multi-layer perceptron on the foundation of NAIS, capturing high-order features by taking the advantages of non-linear hidden layers, which is what NAIS can not achieve. The model can be formulated as:

$$\boldsymbol{v}_{ui} = \sum_{j \in R_u^+ \setminus \{i\}} a_{ij} (\boldsymbol{p}_i \odot \boldsymbol{q}_j) \tag{7}$$

$$\begin{cases} \boldsymbol{v}_1 = ReLU(\boldsymbol{W}_1 \boldsymbol{v}_{ui} + \boldsymbol{b}_1) \\ \boldsymbol{v}_2 = ReLU(\boldsymbol{W}_2 \boldsymbol{v}_1 + \boldsymbol{b}_2) \\ \boldsymbol{v}_L = ReLU(\boldsymbol{W}_L \boldsymbol{v}_{L-1} + \boldsymbol{b}_L) \end{cases} \tag{8}$$

$$\hat{y}_{ui} = \boldsymbol{z}^T \boldsymbol{v}_L + \boldsymbol{b}_u + \boldsymbol{b}_i \tag{9}$$

v_L denotes the output of L^{th} layer, each layer takes the output of last layer as the input, and v_{ui} denotes the second-order

item-item relation. z denotes the weight vector of the prediction layer. Besides, $\boldsymbol{b}_u$ and $\boldsymbol{b}_i$ are introduced to capture the bias of user u's preference and item i's features, respectively. Meanwhile, Deep-ICF has proved that NAIS is a special case of Deep-ICF.

Although these works attempt to model the volatile preferences of users through interacted items, the detailed information about historical items will lose because the features are compressed into a dense vector.

E. MEMORY AUGMENTED NEURAL NETWORKS

Before we elaborate on our proposed model, a brief overview of the memory-based architecture is necessary. Memory augmented neural networks generally consist of a memory module and a controller. The memory module typically contains an external memory encoding long-term information about entities. And the controller performs a series of operations on the memory components such as read, write and erase. The memory module improves model representation capacity independent of the controller while offering an internal representation of knowledge to long-term dependencies.

The initial framework proposed in [43] shows a promising performance to track long-term dependencies in synthetic question answering tasks. However, its strong levels of supervision demand limit its flexibility. End-To-End Memory Networks alleviate the requirement to train the original memory network and become an End-to-End system. Moreover, the attention mechanism of the networks enhances the architecture to focus on specific subsets of most essential information rather than uniformly process all information in a given task.

III. METHODS

In this section, we elaborate on our proposed model Item-based Collaborative Memory Networks(ICMN), Figure 1a shows a visual depiction of the architecture. As can be seen, ICMN maintains two memories: a user historically interacted item memory (long-term memory) and a user preference internal memory (short-term memory), and an embedding vector of the target item. The architecture can remember the detailed high-order item-item similarity and depict the preference of the user depending on the memory module which is capable of encoding the large volume of detailed information. Besides, the neural attention mechanism allows learning a nonlinear weighting function for the historically interacted items, where the more similar items contribute higher weights at the output module. And Figure 1b shows the architecture of multiple hops (layers), which can adaptively learn nonlinear weighting function based on memory.

A. ITEM EMBEDDING

The memory module contains two item-memory matrixes: an internal memory matrix $\boldsymbol{M} \in \mathbb{R}^{Q\times d}$ and an external memory matrix $\boldsymbol{C} \in \mathbb{R}^{Q\times d}$, where Q represents the total number of items and d denotes the size of each memory cell (embedding size). Each historically interacted item j of a user is embedded in two memory slots separately: $\boldsymbol{m}_j \in \boldsymbol{M}$ storing the long-term memory for attributes of the items and $\boldsymbol{c}_j \in \boldsymbol{C}$ storing the short-term memory for features of the items. Each user input is represented by a set of items that the user has interacted with: $\{x_1, x_2, x_3 \ldots x_k\}$, each element in the set denotes a historically interacted item. Then the user input is transformed into a multi-hot encoding. For each multi-hot encoding of a user, we query the internal memory of the historical item $\boldsymbol{m}_j$. Then we project target item input, which is denoted by a one-hot encoding, into an embedding vector $\boldsymbol{e}_i$, which contains the feature information of the target item. Finally, we measure the similarity vector $\boldsymbol{p}'_{ji}$ which consists of the matching information of items as the element-wise product of short-term memory slot $\boldsymbol{m}_j$ and item latent vector $\boldsymbol{e}_i$:

$$\boldsymbol{p}'_{ij} = \boldsymbol{m}_j \odot \boldsymbol{e}_i \tag{10}$$

B. LAYER NORMALIZATION

Due to the large variance on the lengths of user histories [17], the attention mechanism is infeasible to learn from user histories even we utilize the technique mentioned in NAIS which is called Smooth. Specifically, gradients are easily disappearing when we are stacking the hops or enlarging the size of the embedding layer. To resolve the problem, we employ a technique that has been demonstrated effectiveness in the NLP area, namely layer normalization [1]. However, although the vanishing gradient problem disappears, the model losses a bit of capability for representation, which we will introduce in the Experiment section. And later, we will prove the choice that employs the layer normalization in the model is reasonable. Layer normalization is a technique that forces the data to obey the normal distribution and is more suitable for online learning tasks, as well as the learning task that varies with the length of the sequence than batch normalization. We transform the similarity vector by layer normalization as follow:

$$\begin{aligned} \boldsymbol{p}_{ij} &= \frac{\boldsymbol{\gamma}}{\sigma^{ij}} \odot (\boldsymbol{p}'_{ij} - \mu^{ij}) + \boldsymbol{b} \quad \mu^{ij} = \frac{1}{d}\sum_{v=1}^{d} p_{ijv} \\ \sigma^{ij} &= \sqrt{\frac{1}{d}\sum_{v=1}^{d}(p_{ijv} - \mu^{ij})^2} \end{aligned} \tag{11}$$

where μ^{ij}, σ^{ij} are the means and variance of each similarity vector $\boldsymbol{p}_{ij}$. $\boldsymbol{\gamma}, \boldsymbol{b}$ are defined as the bias and gain parameters of the same dimension as $\boldsymbol{p}_{ij}$. p_{ijv} is the elements of $\boldsymbol{p}'_{ij}$ and μ^{ij} is the average of $\boldsymbol{p}'_{ij}$.

C. INTERACTED ITEM ATTENTION

The attention mechanism learns an adaptive weighting function to focus on a subset of most influential items within historical items to predict the interaction. Previous works predefine a uniform weighting function such as inner product and consider all the historical items equally, which may decrease the model fidelity. Therefore, NAIS employs an

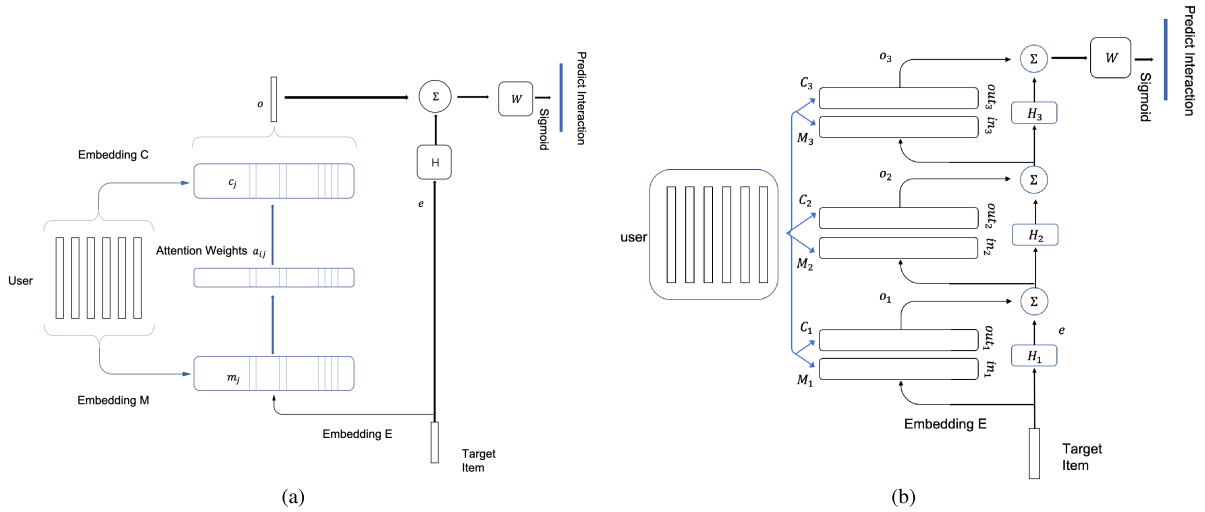

FIGURE 1. ICMN with a single hop(a) and with multiple hops(b).

attention mechanism to learn the most similar neighbor items. However, directly utilizing the standard attention mechanism in the situation where input sequential lengths are varying largely will cause problems, which we have stated in the preliminary part clearly. NAIS has alleviated the problem with a technique called Smooth which has shown effectiveness. Therefore, we introduce this technique to resolve the problem as NAIS did. The attention weights are computed with the following formulation:

$$a_{ij} = \frac{exp(\sum_{v=1}^{d} p_{ijv})}{[\sum_{j} \in R_u^+ \setminus \{i\} exp(\sum_{v=1}^{d} p_{ijv})]^{\beta}} \tag{12}$$

The attention mechanism allows the model to focus on higher specific historical items while paying less attention to historical items which may be less similar. Then we construct the final representation of the user's preference by interpolating the external item memory with the attention mechanism activated by short-term item memory:

$$\boldsymbol{o}_{ui} = \sum_{j \in R_u^+ \setminus \{i\}} a_{ij} \boldsymbol{c}_j \tag{13}$$

where $\boldsymbol{c}_j$ is another embedding vector for item j called external memory, which allows the storage of long-term information about the features of each historical item of users. Intuitively, the internal memory network acts as short-term memory which is activated by the target item, driving the model to search the information about the most similar historical items in the long-term memory then response to the target item. ICMN captures the similarity of items and dynamically assigns the degrees of importance to the historical items instead of uniform weights for historical items which may limit the representation capacity of the model.

D. OUTPUT MODULE

On the foundation of what we constructed before, we perform the prediction for the interaction between user u and target item i. Intuitively, it's inevitable to integrate the information of the target item and the preference of the user to predict the user's response for target items. In other words, we have to combine the embedding vectors $\boldsymbol{e}_i$ which consist of the features of target items and the preference of user $\boldsymbol{o}_{ui}$ which we explored before:

$$\hat{r}_{ui} = \boldsymbol{W}(\boldsymbol{o}_{ui} + \boldsymbol{H}\boldsymbol{e}_i) \tag{14}$$

where $\boldsymbol{e}_i$ denotes the embedding vector of a target item i, and $\boldsymbol{H}$ is a linear transform matrix that can learn with other parameters in the same time. [43]. Our proposed model possesses the following advantages. First, the neural attention mechanism provides adaptive weights for each historical item to the final predicted interaction dependent on the specific item. Second, the memory module integrates the long-term external memory and short-term internal memory for each historical item, and the associative contribution of both memory components provides more complete historical information about the items for the model to achieve accurate prediction.

E. MULTIPLE HOPS

We now extend the model to tackle an arbitrary number of memory hops. Figure 1b shows the ICMN's architecture with multiple hops. The initial hop introduces the demand to acquire additional information of items. Starting from the second hop, the model begins to consider the high-order item similarity. Each additional hop handles the last hop's newly obtained information. In other words, the model has chances to retrospect and reconsider the most similar historical items. Specifically, multiple memory modules are stacked together by taking the output from the l^{th} hop as input to the $(l+1)^{th}$ hop:

$$\boldsymbol{e}_i^{l+1} = (\boldsymbol{H}^l \boldsymbol{e}_i^l + \boldsymbol{o}_{ui}^l) \tag{15}$$

where $\boldsymbol{H}^l \in \mathbb{R}^{d \times d}$ is a linear project matrix mapping the target item's feature to a latent space coupled with the existing information from the last hop [10]. The newly formed

target item attribution vector then awakes the short-term item memory and recomputes the adaptive attention weights. This process is repeated for each hop generating an iterative refinement. The output module receives the final weighted neighborhood items from the last hop to predict the final interaction. We employ the Layer-wise type of weight tying within the model [36]. That is to say, the input and output embeddings are the same across different layers, which can reduce massive space complexity and time complexity:

$$\begin{aligned} \boldsymbol{C}_1 &= \boldsymbol{C}_2 = \cdots = \boldsymbol{C}^k \\ \boldsymbol{M}_1 &= \boldsymbol{M}_2 = \cdots = \boldsymbol{M}^k \end{aligned} \tag{16}$$

F. OPTIMIZATION

To optimize our model, we should build an objective function to minimize the error between ground-truth interaction and the interaction predicted by the model. Due to the implicit feedback where each entry is a binary value 0 or 1, we can view the learning task as a binary classification task. As the previous [18] did, we treat the observed user-item interaction as a positive instance and sample negative instances from the remaining unobserved interactions. Meanwhile, we let R^+ and R^- denote the sets of positive and negative instances respectively, and the loss function is defined as follows:

$$\begin{aligned} L = -\frac{1}{N}(&\sum_{(u,i\in R^+)} \log \sigma(\hat{r}_{ui}) \\ &- \sum_{(u,i\in R^-)} \log \sigma(1-\hat{r}_{ui}) + \lambda||\theta||^2) \end{aligned} \tag{17}$$

where N denotes the total number of training instances. The λ is a hyper-parameter that prevents the model from overfitting, θ denotes all the trainable parameters. And σ is a sigmoid function that restricts the prediction $\hat{r}_{ui}$ in the range of [0,1]. Based on this, the prediction can be transformed into a probability that represents the likelihood that the user will interact with the item. It is obvious that the objective function we constructed is same with binary cross-entropy loss. There are also other options of objective functions, such as the pointwise regression [19], [41] and pairwise ranking losses [30], [50], suitable to learn ICMN with implicit feedback. Considering the focus of this work is to research employing memory network to item-ICF, we leave the exploration of other loss functions as future works. For convenience, we adopt Adam [22] to optimize the objective function, which updates the learning rate automatically based on adaptive estimates of lower-order moments to alleviate the pain of tuning the learning rate.

IV. EXPERIMENT

In this section, we show a series of experimental results and analyze them.

A. EXPERIMENTAL SETTINGS

1) DATASETS

We experimented on two well-known datasets: the MovieLens and the Pinterest. The characteristics of the two datasets are summaries in Table 1. More details about the datasets have been elaborated in [18] and we do not restate it. Due to both datasets have been preprocessed such as removing sparse users and train-test splitting, we directly utilize the preprocessed data to train and test our model. What is worth to note is that each positive instance is paired with 4 negative instances so the total number of training instances is five times the number of interactions.

TABLE 1. Characteristics of the evaluation datasets.

Dataset	Interaction#	Item#	user#	Sparsity
MovieLens	1,000,209	3,706	6,040.	95.53%
Pinterest	1,500,809	9,916	55,187	99.73%

2) EVALUATION PROTOCOLS

We adopt the leave-one-out evaluation protocol [18], [30], which is popular in implicit feedback based collaborative filtering. It reserves the latest interaction as the validation data, the rest are fed to training. We randomly sample 99 unobserved items (negative instances) for each user, pairing it with the validation data as the test data set; then feed the test data set to each method and get prediction scores for the 100 instances. Finally, Hit Ratio (HR) [9] and Normalized Discounted Cumulative Gain (NDCG) [15] are chosen to be the evaluation criteria. Here, we shortly brief the two metrics: HR can be deemed as a recall-based measure that estimates the percentage of users who are successfully recommended by observing whether the positive instance present at the top-10 rank list, and NDCG measures the quality of ranking, which assigns higher scores to hit at top position ranks. That means the higher values of both metrics are, the better performance is. Both metrics are widely used in ranking systems [16].

3) BASELINES

We compare ICMN with the following recommendation methods:

- Pop. This is a non-personalized method to benchmark the performance of the top-K recommendation task. It ranks items by their popularity, judging by the number of interactions that an item received.
- ItemKNN [32]. This is the standard item-based CF method we mentioned in Equation 1. We used consine similarity to measure s_{ij}; and followed the setting of [17] to adapt it.
- HOSLIM [4]. This model extends SLIM, leraning two sparse aggregation coefficient matrices to capture higher-order item relations.
- FISM [20]. This is a learning based item-based CF model as formulated in Equation 3. We set α to 0.1 which leads to best result on both dataset [17].
- Youtube Rec [6]. A deep neural network architecture for recommending YouTube videos. It maps video IDs to a sequence of embeddings and feed them into a feed-forward neural network. The input layer is followed

by several layers of fully connected hidden layer and the nerual is activated by Rectified Linear Units (ReLU).

- MF-BPR [30]. MF-BPR optimizes MF with the pair-wise Bayesian Personalized Ranking BPR) loss. It is a highly popular baseline for item recommendation. We optimized it with fixed learning rate, and reported the best performance.
- MF-eALS [19]. MF-eALS also learns a MF model, but optimizes a different pointwise regression loss which called element-wise Alternating Learning Square (eALS) algorithm, treating all unobserved interactions as negative feedback with a smaller weight.
- MLP [18]. This method learns the scoring function from user-item interacions based on the effectiveness of multi-layers perception. We employed a 3-layer MLP and optimized the same cross-entropy loss, which was illustrated to perform well on the two datasets.
- Deep-ICF [46]. Considering that NAIS is a special case of Deep-ICF, to prove the effectiveness of large memory in collaborative filtering, we chose Deep-ICF to compare with our model rather than NAIS. Meanwhile, We tuned the parameters of Deep-ICF as same as the original literature.

B. PARAMETER SETTINGS

We implemented our proposed methods based on Keras and ran the process on GPU. To simplicity, without special mention, we set the hyper-parameter β to 0.5 and sampled four instances per positive instance. Besides, We split the training instance with a fixed mini-batch size of 4096 on the Pinterest dataset, while splitting the training instance with user-based mini-batch on the MovieLens Dataset, since masking larger variance of the historical item length on the MovieLens cost quite much time, but on the Pinterest, the variance is relatively smaller and the number of users is massive which will consume time to transmit data from CPU to GPU. Moreover, for ICMN models that are trained from scratch, we initialized model parameters with a Xavier normal distribution. We tested the learning rate of [0.001,0.0005, 0.0001], embedding size of [8, 16, 32, 64], l_2 regularization of embedding matrix in the range of $[1^{-7}, 1^{-6}, \ldots\ldots, 1^{-3}]$, and l_2 regularization of hidden layer and gain parameter in the range of $[1^{-4}, 1^{-3}, \ldots\ldots, 1]$.

C. EFFECT OF PRE-TRAINING

Due to the non-linearity of the neural attention mechanism, the complexity of the memory network, and the non-convexity of the objective function, optimizing the model from scratch can be easily trapped to local minimums of poor performance. Therefore, the initialization of model parameters plays a crucial role in the model's final performance. Intuitively, optimizing memory module and neural attention mechanism simultaneously is difficult, as such, we pretrained the ICMN/w which is ICMN without attention mechanism (without layer normalization), and used the pretrained weights to initialize ICMN.

TABLE 2. Performance of ICMN with and without pre-training.

	With Pretraining		Without Pretraining	
Hops	HR@10	NDCG@10	HR@10	NDCG@10
MovieLens				
1	**71.56**	**44.53**	70.76	43.35
2	**72.77**	**44.69**	71.03	43.45
3	**71.71**	**44.34**	71.21	43.83
4	**71.11**	**43.80**	70.69	43.74
Pinterest				
1	**88.63**	**56.66**	87.79	55.36
2	**88.90**	**57.55**	87.90	55.51
3	**88.94**	**57.78**	88.12	56.27
4	**88.58**	**57.46**	88.48	57.17

To demonstrate the effect of pre-training (*i.e.*, using the parameter learned by the model without attention as ICMN's initialization), table 2 shows the performance of ICMN with and without pre-training at embedding size 64. It's obvious that the ICMN improves significantly with the pre-trained parameters. Moreover, as our observation, ICMN with pre-training converges faster than the one with random initializations. To show the best performance of our proposed ICMN, without special mention, we use pre-training tricks to optimize the model.

D. EMBEDDING SIZE AND THE NUMBER OF HOPS

Since we mentioned that stacking hops can help the back hops utilize the information that previous hops captured and the neural attention recomputes the adaptive weights for items. Moreover, it's obvious that embedding size is vital for the model since a larger embedding size vector provides more features of an item. Now we explore the effect of embedding size and the number of hops. Table 3, 4 shows the performance of ICMN with embedding size of [8, 16, 32, 64] and the number of hops of [1, 2, 3, 4]. As can be observed: on the Pinterest dataset, with the increasing of embedding size and the number of hops, the performance of ICMN is increasingly improving, while on the MovieLens dataset, with the increasing of embedding size, the performance is generally better, but is up and down as stacking the hops in large embedding size. The reason is that the MovieLens dataset lacks enough data to train the ICMN and the model becomes overfitting.

E. EFFECTIVENESS OF ATTENTION NETWORKS WITH LAYER NORMALIZATION

As the effectiveness of only neural attention networks in the collaborative filtering model has been illustrated in the literature [17], we do not restate it. We mentioned before that layer normalization may limit the representation for an entity, and now we illustrate it here: layer normalization can force the data flowing in the model to obey normal distribution to accelerate the speed of training and prevent the model from vanishing/exploding gradient. However, restricting the features of items to be standard normal distribution is not a wise selection. For example, a dimension of embedding vector of an item represents that the item services for female

TABLE 3. HR@10 in various embedding sizes and number of hops.

Factors	ICMN-1	ICMN-2	ICMN-3	ICMN-4
MovieLens				
8	66.72	66.94	66.97	**67.40**
16	68.66	70.07	70.36	**70.60**
32	71.49	71.44	**71.56**	71.11
64	71.56	**72.77**	71.71	71.11
Pinterest				
8	87.20	87.25	87.36	**87.56**
16	88.06	88.17	88.21	**88.28**
32	88.12	88.31	88.20	**88.86**
64	88.63	88.90	**88.94**	88.58

TABLE 4. NDCG@10 in various embedding sizes and numbers of hops.

Factors	ICMN-1	ICMN-2	ICMN-3	ICMN-4
MovieLens				
8	39.64	39.72	39.96	**40.24**
16	42.22	42.45	42.89	**43.33**
32	43.77	43.60	**43.84**	43.72
64	44.53	**44.69**	44.34	43.80
Pinterest				
8	54.65	55.04	55.13	**55.47**
16	56.14	56.58	56.71	**56.82**
32	56.03	56.63	56.86	**57.64**
64	56.66	57.55	**57.96**	57.46

TABLE 5. Performance of ICMN with and without attention in embedding size of 16.

	With Attention		Without Attention	
Hops	HR@10	NDCG@10	HR@10	NDCG@10
MovieLens				
1	**68.66**	**42.22**	68.41	41.22
2	**69.14**	**42.45**	69.01	41.38
3	**70.46**	**42.89**	68.91	41.44
4	**70.69**	**43.33**	69.19	41.61
MovieLens				
1	**88.06**	56.14	87.42	**56.20**
2	**88.17**	**56.58**	87.79	56.10
3	**88.21**	**56.71**	87.86	56.22
4	**88.28**	**56.82**	87.95	56.45

or male; obviously, it's a binary value, and another dimension represents the age of client that the item services; clearly, forcing the two dimensions to obey the same normal distribution will cause loss of information. However, directly employing the attention mechanism makes us be at a loss for the vanishing gradient problem (the result of our experiments). Thus, we analyze whether we should employ the attention mechanism with layer normalization or not. Table 5 shows the performance of the model with attention (with layer normalization) and the model without it in the embedding size of 16. As can be seen, the model employing attention with layer normalization achieves better performance. Although on the Pinterest dataset, with one hop, the model with attention performs slightly worse than the one without attention because of losing the capacity of representation but performs better when we are stacking hops as the advantage of reconsidering the attention covers the limitation. Besides, with the increasing number of hops, the model improves slightly without the attention mechanism but improves significantly with the assistance of the attention mechanism.

TABLE 6. Performance of HR and NDCG of compared methods at embedding size 16.

	MovieLens		Pinterest	
Methods	HR@10	NDCG@10	@HR@10	NDCG@10
Pop	45.13	25.51	27.35	14.09
ItemKNN	62.24	35.81	78.51	48.24
HOSLIM	68.51	42.33	86.42	55.43
MF-BPR	66.32	39.05	86.33	53.95
Youtube Rec	68.63	42.76	86.21	53.89
MF-eALS	66.57	39.42	87.23	52.08
MLP	68.39	40.86	85.96	52.68
FISM	66.72	39.39	86.89	55.14
Deep-ICF	68.31	41.07	88.10	56.08
ICMN	**70.6**	**43.33**	**88.28**	**56.82**

1) SMOOTHING EXPONENT β

In this section, we study the performance of varying the smoothing exponent β for ICMN reporting HR@10 and NDCG@10. We collected the result of three hops. Figure 3 shows the performance of ICMN with different smoothing exponent β varying from 0 - 1 on both datasets. We can see that when β is smaller than 1 the performances of ICMN is acceptable. However, when β is set to 1, the performance of ICMN degrades rapidly. The result is consistent with the conclusion of NAIS [17] and the issue is caused by the large variance of the length of user histories.

2) BASELINE COMPARISON

Now, we compare the performance of ICMN with other state-of-the-art collaborative filtering methods. We first make a comparison with other recommendation approaches in the embedding size of 16 to analyze the performance of all methods. Next, we alter the embedding size to observe varying embedding size trends. Table 6 shows the overall recommendation accuracy. Through the observation, we can easily acquire the following conclusion:

1) ICMN achieves the best performance on both datasets (the highest HR and NDCG scores) We attribute improvements to the effect of the adaptive neural attention weights for historical items, and the powerful combined action of both long-term and short-term memory capacity.
2) There is no doubt that learning-based CF methods perform better than heuristic-based approaches such as Pop and ItemKNN. Comparing FISM with ItemKNN, both methods are item-based CF, we can readily see the advantages of learning-based methods over traditional statistical methods.
3) There is no overall winner between the user-based and item-based CF models. In particular, user-based models perform better than FISM on the MovieLens, while worse on Pinterest. We conclude that item-based CF methods are more advantageous on a highly sparse dataset [20].

Figure 4 illustrates the score of HR and NDCG for the embedding size on the two datasets. Obviously, the tendencies of performance at embedding size of 8, 32, and

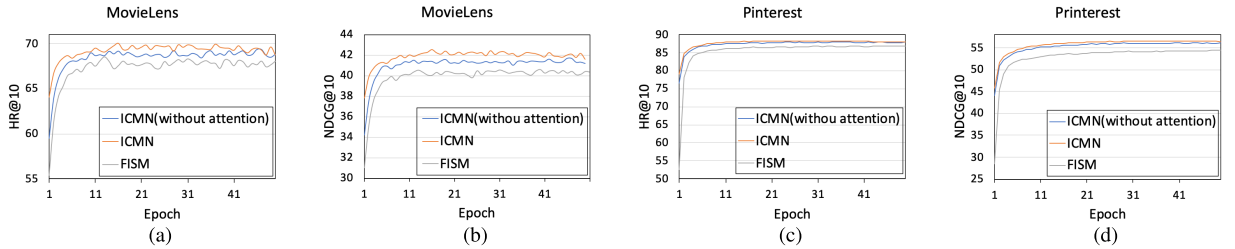

FIGURE 2. Performance of FISM, ICMN/w, ICMN in each epoch.

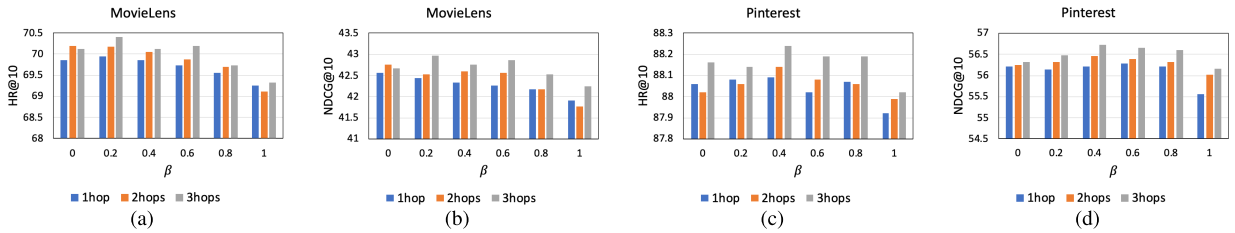

FIGURE 3. Evaluation of ICMN methods *w.r.t.* the smoothing exponent β.

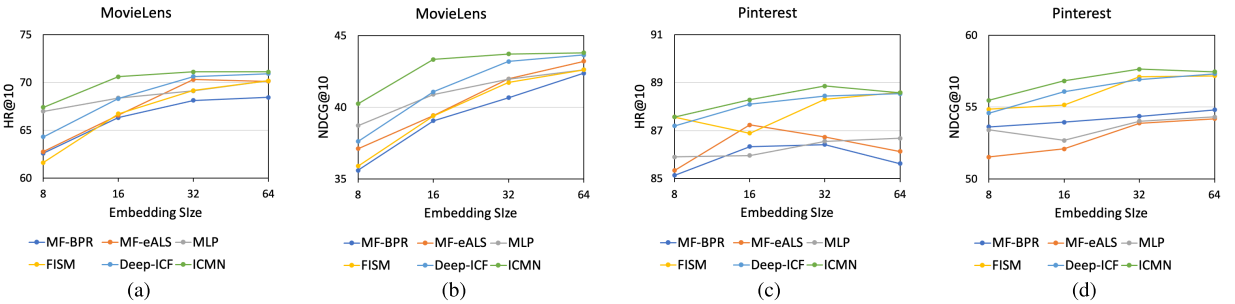

FIGURE 4. Performance of compared methods *w.r.t.* embedding size.

64 are similar to the one of 16, generally. Our ICMN approaches outperform all the other approaches.

Figure 2 shows the performance in the first 50 epochs of ICMN, ICMN/w, and FISM at embedding size 16 on the two datasets. First, we can see that each of the three ICF models shows a smooth curve on the Pinterest but an uneven curve on the MovieLens. We argue the reason is the different variance of the user's historical item length in the two datasets. [17]. Second, among the three approaches, ICMN achieves the best result followed by ICMN/w and then FISM, we attribute the improvement of ICMN/w over FISM to the cooperation of the two memory components and the further improvements of ICMN over ICMN/w to the advanced neural attention mechanism.

The above findings provide empirical evidence for the effectiveness of the rationality of memory networks for ICF. However, a massive number of parameters of the architecture makes the model easily overfit, we will take measures to tackle the issue in future work.

V. RELATED WORK

Early works on recommendation primarily focus on explicit feedback [31], [32], which is to minimize the loss between observed ratings and the predicted ratings by the corresponding model. Typically, the well-known regression-based CF method MF, associating each user an item with a latent vector, modeling the rating score of the user by the inner product of the vectors, achieved the best performance in the Netflix challenge. Literature [24], [25], [27], [34], [38], [42] extended MF with additional information such as review texts and social influence.

Later, a surge of interest in recommendation with implicit feedback has emerged [2], [5], [11], [12], [15]. While implicit feedback is more challenging to utilize as user satisfaction can not be observed and the natural scarcity of negative feedback, implicit feedback is much easier to collect for content providers since it indirectly reflects users' preference through behaviors such as clicking items, watching videos, and reading news.

Recently, literature about applying deep neural networks (DNN) for recommendation springs up. These DNN models construct a strong representation of the entity content. Neural Collaborative Filtering [18] deals with implicit feedback by learning a model associating a matrix factorization and a deep feedforward neural network. The prediction is based on linear information and non-linear relation.

Cheng *et al.* [3] jointly trained wide linear models and DNN to combine the benefits of memorization and generalization for recommender systems. Convolutional neural networks (CNN) are also popular in recommendation literature, which has been used to capture local feature representations of entities such as images, [48], text [5], [21], music [8], and so on. Besides, Attention mechanisms also have been focused on recommendation systems. Gong and Zhang [14] perform hashtag recommendation with CNN and the attention mechanism to focus on the most informative words. NAIS [17] develops a smooth technique to make the attention mechanism suitable for variant sequence length, which inspires us to propose our method.

The most similar method to ours is the Collaborative Memory Network (CMN) [36], which combines model-based CF and neighborhood-based CF with End-to-End Memory Network. Our methods differ from CMN and all previous works by employing a memory network and attention mechanism with layer normalization for item-based CF. Besides, we introduce the technique that smooths the denominator of the *softmax* function due to the standard attention network does not perform well on user historical interaction.

To the best of our knowledge, there is no prior work that has applied the End-to-End memory network architecture to deal with Item-based collaborative filtering and our experimental results show that Memory networks is a promising choice to track user tastes on different attributes of items.

VI. CONCLUSION

In this work, we explored memory network architectures for item-based collaborative filtering. Our key argument is that users' past behavior will influence the present behavior, and long-term memory for user's behavior is vital for prediction but hard to record. Besides, learning the attention network from largely varying historical item length the large variance on the lengths of user histories will cause problems. Therefore, we adopted smooth and layer normalization into our method to enhance the robustness of the model and achieved a promising effect. As can be seen, we proved the models with layer normalization and attention beyond the models without them. We also showed the salient effect of the models with a strong memory and illustrated the effectiveness of the smoothing exponent β.

In the future, we will explore new technologies to resolve the vanishing gradient problem without losing representation. Besides, although the result is promising, the model exists the problem of overfitting since the model has a huge number of parameters. We will investigate more mathematical methods to handle this issue.

REFERENCES

[1] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*. [Online]. Available: https://arxiv.org/abs/1607.06450

[2] I. Bayer, X. He, B. Kanagal, and S. Rendle, "A generic coordinate descent framework for learning from implicit feedback," in *Proc. 26th Int. Conf. World Wide Web*, Apr. 2017, pp. 1341–1350.

[3] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah, "Wide & deep learning for recommender systems," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, 2016, pp. 7–10.

[4] E. Christakopoulou and G. Karypis, "HOSLIM: Higher-order sparse linear method for top-N recommender systems," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining*, 2014, pp. 38–49.

[5] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, Jan. 2008, pp. 160–167.

[6] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for YouTube recommendations," in *Proc. 10th ACM Conf. Recommender Syst.*, Sep. 2016, pp. 191–198.

[7] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and D. Sampath, "The YouTube video recommendation system," in *Proc. 4th ACM Conf. Recommender Syst.*, 2010, pp. 293–296.

[8] A. van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 2643–2651.

[9] M. Deshpande and G. Karypis, "Item-based top-N recommendation algorithms," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 143–177, Jan. 2004.

[10] T. Ebesu, B. Shen, and Y. Fang, "Collaborative memory network for recommendation systems," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jun. 2018, pp. 515–524.

[11] A. M. Elkahky, Y. Song, and X. He, "A multi-view deep learning approach for cross domain user modeling in recommendation systems," in *Proc. 24th Int. Conf. World Wide Web (WWW)*, 2015, pp. 278–288.

[12] X. Geng, H. Zhang, J. Bian, and T.-S. Chua, "Learning image and user features for recommendation in social networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4274–4282.

[13] C. Gomezuribe and N. D. Hunt, "The Netflix recommender system: Algorithms, business value, and innovation," *ACM Trans. Manage. Inf. Syst.*, vol. 6, no. 4, p. 13, 2016.

[14] Y. Gong and Q. Zhang, "Hashtag recommendation using attention-based convolutional neural network," in *Proc. 25th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2016, pp. 2782–2788.

[15] X. He, T. Chen, M.-Y. Kan, and X. Chen, "TriRank: Review-aware explainable recommendation by modeling aspects," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2015, pp. 1661–1670.

[16] X. He, M. Gao, M.-Y. Kan, and D. Wang, "BiRank: Towards ranking on bipartite graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 1, pp. 57–71, Jan. 2017.

[17] X. He, Z. He, J. Song, Z. Liu, Y.-G. Jiang, and T.-S. Chua, "NAIS: Neural attentive item similarity model for recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 12, pp. 2354–2366, Dec. 2018.

[18] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.

[19] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2016, pp. 549–558.

[20] S. Kabbur, X. Ning, and G. Karypis, "FISM: Factored item similarity models for top-N recommender systems," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2013, pp. 659–667.

[21] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proc. 10th ACM Conf. Rec. Syst.*, Sep. 2016, pp. 233–240.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980

[23] S. Li, J. Kawale, and Y. Fu, "Deep collaborative filtering via marginalized denoising auto-encoder," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2015, pp. 811–820.

[24] D. Lian, K. Zheng, Y. Ge, L. Cao, E. Chen, and X. Xie, "GeoMF++: Scalable location recommendation via joint geographical modeling and matrix factorization," *ACM Trans. Inf. Syst.*, vol. 36, no. 3, p. 33, 2018.

[25] Y. Liao, W. Lam, L. Bing, and X. Shen, "Joint modeling of participant influence and latent topics for recommendation in event-based social networks," *ACM Trans. Inf. Syst.*, vol. 36, no. 3, p. 29, 2018.

[26] D. C. Liu, S. Rogers, R. Shiau, D. Kislyuk, K. C. Ma, Z. Zhong, J. Liu, and Y. Jing, "Related pins at pinterest: The evolution of a real-world recommender system," in *Proc. 26th Int. Conf. World Wide Web Companion (WWW Companion)*, 2017, pp. 583–592.

[27] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: Understanding rating dimensions with review text," in *Proc. 7th ACM Conf. Recommender Syst. (RecSys)*, 2013, pp. 165–172.
[28] X. Ning and G. Karypis, "SLIM: Sparse linear methods for top-N recommender systems," in *Proc. IEEE 11th Int. Conf. Data Mining*, Dec. 2011, pp. 497–506.
[29] Z. Ren, S. Liang, P. Li, S. Wang, and M. D. Rijke, "Social collaborative viewpoint regression with explainable recommendations," in *Proc. 10th ACM Int. Conf. Web Search Data Mining (WSDM)*. New York, NY, USA: Association for Computing Machinery, 2017, pp. 485–494.
[30] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidtthieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. 25th Conf. Uncertainty Artif. Intell.*, 2009, pp. 452–461.
[31] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering," in *Proc. 24th Int. Conf. Mach. Learn. (ICML)*, 2007, pp. 791–798.
[32] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web (WWW)*, 2001, pp. 285–295.
[33] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 111–112.
[34] L. Shi, W. X. Zhao, and Y. Shen, "Local representative-based matrix factorization for cold-start recommendation," *ACM Trans. Inf. Syst.*, vol. 36, no. 2, p. 22, 2017.
[35] B. Smith and G. Linden, "Two decades of recommender systems at Amazon.com," *IEEE Internet Comput.*, vol. 21, no. 3, pp. 12–18, May 2017.
[36] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, "End-to-end memory networks," in *Advances in Neural Information Processing Systems*, vol. 28, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, Inc., 2015, pp. 2440–2448. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/8fb21ee7a2207526da55a679f0332de2-Paper.pdf
[37] C. Sun, H. Liu, M. Liu, Z. Ren, T. Gan, and L. Nie, "LARA: Attribute-to-feature adversarial learning for new-item recommendation," in *Proc. 13th Int. Conf. Web Search Data Mining (WSDM)*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 582–590.
[38] Y. Sun, N. J. Yuan, X. Xie, K. R. Mcdonald, and R. Zhang, "Collaborative intent prediction with real-time contextual data," *ACM Trans. Inf. Syst.*, vol. 35, no. 4, p. 30, 2017.
[39] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," 2014, *arXiv:1409.2944*. [Online]. Available: https://arxiv.org/abs/1409.2944
[40] H. Wang, N. Wang, and D. Yeung, "Collaborative deep learning for recommender systems," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 1235–1244.
[41] M. Wang, W. Fu, S. Hao, H. Liu, and X. Wu, "Learning on big graph: Label inference and regularization with anchor hierarchy," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 5, pp. 1101–1114, May 2017.
[42] X. Wang, X. He, L. Nie, and T. Chua, "Item silk road: Recommending items from information domains to social users," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2017, pp. 185–194.
[43] J. Weston, S. Chopra, and A. Bordes, "Memory networks," 2015, *arXiv:1410.3916*. [Online]. Available: https://arxiv.org/abs/1410.3916
[44] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, "Recurrent recommender networks," in *Proc. 10th ACM Int. Conf. Web Search Data Mining (WSDM)*, 2017, pp. 495–503.
[45] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-N recommender systems," in *Proc. 9th ACM Int. Conf. Web Search Data Mining (WSDM)*, 2016, pp. 153–162.
[46] F. Xue, X. He, X. Wang, J. Xu, K. Liu, and R. Hong, "Deep item-based collaborative filtering for top-N recommendation," *ACM Trans. Inf. Syst.*, vol. 37, no. 3, pp. 1–25, Jul. 2019.
[47] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, and T. Chua, "Discrete collaborative filtering," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2016, pp. 325–334.
[48] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, p. 5, 2019.
[49] S. Zhang, L. Yao, and X. Xu, "AutoSVD++: An efficient hybrid collaborative filtering model via contractive auto-encoders," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2017, pp. 957–960.
[50] Z. Zhao, B. Gao, V. W. Zheng, D. Cai, X. He, and Y. Zhuang, "Link prediction via ranking metric dual-level attention network learning," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 3525–3531.

DEWEN SENG received the Ph.D. degree from the University of Science and Technology, Beijing, China, in 2005. He is currently an Associate Professor and the Vice Director with the Software Engineering Institute, Hangzhou Dianzi University. He has (co) authored more than 40 articles. His main current research interests include artificial intelligence and data mining technology.

GUANGSEN CHEN is currently pursuing the master's degree with the School of Computer, Hangzhou Dianzi University. His current research interests include recommendation systems and computer vision.

QIYAN ZHANG is currently pursuing the master's degree with the School of Computer, Hangzhou Dianzi University. Her current research interests include deep learning frameworks and data mining techniques.