Matching User with Item Set: Collaborative Bundle Recommendation with Deep Attention Network

Liang Chen[1,2], **Yang Liu**[1,2], **Xiangnan He**[3], **Lianli Gao**[4] and **Zibin Zheng**[1,2*]
[1]School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China
[2]Guangdong Key Laboratory for Big Data Analysis and Simulation of Public Opinion, The school of Communication and Design, Sun Yat-sen University, Guangzhou, China
[3]University of Science and Technology of China
[4]The University of Electronic Science and Technology of China
chenliang6@mail.sysu.edu.cn, liuy296@mail2.sysu.edu.cn, xiangnanhe@gmail.com, lianli.gao@uestc.edu.cn, zhzibin@mail.sysu.edu.cn

Abstract

Most recommendation research has been concentrated on recommending single items to users, such as the considerable work on collaborative filtering that models the interaction between a user and an item. However, in many real-world scenarios, the platform needs to show users a set of items, e.g., the marketing strategy that offers multiple items for sale as one bundle. In this work, we consider recommending a set of items to a user, i.e., the *Bundle Recommendation* task, which concerns the interaction modeling between a user and a set of items. We contribute a neural network solution named DAM, short for *Deep Attentive Multi-Task model*, which is featured with two special designs: 1) We design a factorized attention network to aggregate the item embeddings in a bundle to obtain the bundle's representation; 2) We jointly model user-bundle interactions and user-item interactions in a multi-task manner to alleviate the scarcity of user-bundle interactions. Extensive experiments on a real-world dataset show that DAM outperforms the state-of-the-art solution, verifying the effectiveness of our attention design and multi-task learning in DAM.

1 Introduction

Recommender system [Xie *et al.*, 2018a; Xie *et al.*, 2018b] has long been an effective tool to alleviate the information overload, to improve user experience, and to increase the traffic of service provider [Smith and Linden, 2017]. Similar to classic techniques in information retrieval, recommendation is typically cast as a matching problem [Xu *et al.*, 2018] and solved by estimating the matching score between a user and an item [He *et al.*, 2017]. Nevertheless, many real-world applications need to recommend a set of items for users to consume **as a whole**, such as travel package [Ge *et al.*, 2014], music playlist [Cao *et al.*, 2017], furniture set, and so on. We term all such scenarios that concern recommending a set of items as *Bundle Recommendation*, which needs to predict a user's preference on a bundle of items rather than a single item [Zhu *et al.*, 2014].

Figure 1: Example of product bundles (relevant songs and complementary electronic products).

In E-commerce, product bundling is a widely used strategy to support promotional campaigns — e.g., purchasing a bundle has a discounted rate — which, in turn, can increase the exposure of items that are seldom purchased in isolation. To build a bundle recommendation system, a common solution pipeline is to first generate bundles based on some criteria (e.g., products that form a complementary relation [Wang *et al.*, 2018b] as shown in Figure 1), and then build models to predict user preference on bundles [Pathak *et al.*, 2017]. When a bundle recommendation system is brought online to serve users for a period, the platform can accumulate user behavior logs on bundles, offering opportunities to improve the bundle recommendation service. In this work, we focus on the second phase of building user preference models on bundles, assuming that there are a certain number of user-bundle interactions available to indicate user preference on bundles.

Intuitively, this problem can be solved by collaborative filtering (CF) methods by treating bundles as "items" in traditional CF modeling. Although technically feasible, such straightforward solutions do not work well to predict user-bundle interactions due to the following difficulties:

- **Bundles are not atomic units.** Fundamentally different from items in traditional CF, a bundle is not an atomic unit since it is composed of multiple (at least two) items. Intuitively, bundles that share more items should exhibit

*Contact Author

similar patterns in attracting users. As such, it is inappropriate to treat bundles as separate columns in user-bundle matrix to run CF methods like matrix factorization.

- **User-bundle interactions are more sparse.** Only when a user is mostly satisfied with items in a bundle, she will choose the bundle to consume. Sometimes even if a user likes all constituent items, she may dislike the bundle because it is not a good match. As such, user-bundle interactions are usually more sparse than user-item interactions. Moreover, it is easy to form new bundles thus the cold-start issue is more common in user-bundle data.

In this work, we contribute a new solution for collaborative bundle recommendation named *Deep Attentive Multi-Task* (DAM) model, which takes special care of the above-mentioned two difficulties with neural network techniques:

- **Handling non-atomic bundles.** To account for the compositional similarity between bundles, we aggregate item embeddings to form the bundle representation, rather than associating a bundle ID with an embedding. Such a mechanism equips our method with the natural ability to handle cold-start bundles. Moreover, we design a factorized attention network to take in user preference on constituent items to represent the bundle, which captures the intuition that different users may like the same bundle for different reasons.
- **Handling sparse user-bundle interactions.** To alleviate the sparsity of user interactions on bundles, we integrate user-items interactions which provide additional CF signal on user interests and item properties. In DAM model design, we share user embeddings to predict both user-bundle interactions and user-item interactions, and share item embeddings to learn bundle representations and to predict user-item interactions. During model learning, we tightly couple the models for user-bundle and user-item interactions in a multi-task manner [Yosinski *et al.*, 2014], allowing the benefits of one task (user-item modeling) to be transferred to another task (user-bundle modeling).

2 Problem Formulation

First we introduce the notation conventions. We use bold uppercase letters to denote matrices (e.g., $\mathbf{W}$), bold lowercase letters to denote vectors (e.g., $\mathbf{w}$), and non-bold letters to denote scalars or indices (e.g., w). The uppercase calligraphic symbols (e.g., $\mathcal{W}$) stand for sets.

Suppose we have N users $\mathcal{U} = \{u_1, u_2, ..., u_N\}$, M items $\mathcal{V} = \{v_1, v_2, ..., v_M\}$, and K bundles $\mathcal{B} = \{b_1, b_2, ..., b_K\}$; we use i, j, and s to represent the ID of a user, an item, and a bundle, respectively. Each bundle b_s is also represented as a set of items, denoted as $\mathcal{G}_s = \{g_{s,1}, g_{s,2}, ..., g_{s,|b_s|}\}$, where $|b_s|$ denotes the bundle size (larger than 1), and each item in the bundle belongs to the set $\mathcal{V}$. A user can have an interaction (e.g., click, purchase, or review) on an item or a bundle, which is a binary variable — 1 means the existence of the interaction and 0 otherwise. We use $\mathbf{H} = [h_{ij}]_{N \times M}$ to denote the user-item interaction matrix and $\mathbf{R} = [r_{is}]_{N \times K}$ to denote the user-bundle interaction matrix. Then, we formulate the collaborative bundle recommendation task as:

Input: Users $\mathcal{U}$, items $\mathcal{V}$, bundles $\mathcal{B}$, constituent items of bundles $\{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_K\}$, user-item interactions $\mathbf{H}$, and user-bundle interactions $\mathbf{R}$.

Output: A personalized scoring function that maps a bundle $\mathcal{G}$ to a real value for each user: $f_u : \mathcal{G} \rightarrow \mathbb{R}$.

Note that cold-start bundles are common in bundle recommendation scenarios, for example, service provider launches new promotion packages to customers. As such, it is not necessary that the bundle $\mathcal{G}$ inputted to the personalized scoring function f_u has appeared in $\mathcal{B}$, which contains historical bundles in training set only. This requires the model to score a bundle based on its constituent items rather than the bundle's ID, making the problem radically different from traditional recommendation that scores each single item.

3 Methodology

Our proposed DAM method aims to learn the personalized bundle ranking function from user-bundle interactions $\mathbf{R}$ by properly incorporating user-item interactions $\mathbf{H}$. There are two key designs in DAM: 1) bundle representation learning which aims to obtain the latent features to represent a bundle, and 2) jointly modeling of user interactions on bundles and items. This section is organized to elaborate the two parts.

3.1 Bundle Representation Learning

In terms of provided data, a bundle is described by an ID and its constituent items. However, it is prohibitive to use ID to encode a bundle, since it will make the model less generalizable to new bundles. As such, we consider representing a bundle based on its constituent items only. In the paradigm of neural network, an entity of interest is typically represented by a real-valued vector, also called as *embedding* or *latent features*; then the model is operated on the vector to make predictions. Here we associate each item v_j with an embedding $\mathbf{v}_j$, directly projecting item one-hot ID to the latent space. Next, we aggregate the item embeddings in a bundle to obtain the bundle's embedding.

There are several predefined strategies in neural networks to aggregate embeddings, such as concatenation [Cheng *et al.*, 2016], average pooling, max pooling [Wang *et al.*, 2018a], and bilinear interaction pooling [He and Chua, 2017]. Technically speaking, concatenation is not applicable here, since different bundles may have different sizes thus concatenation may lead to variable-length vector for different training examples. While the three pooling strategies can be applied to convert a vector set to one vector, they cannot capture the intuition in bundle recommendation. Intuitively, different items play different roles in a bundle; e.g., for the first bundle shown in Figure 1, the Mac plays a dominant role since it is much more expensive than other accessories. Moreover, the weighting scheme can be varied for different users; e.g., for the second bundle shown in Figure 1, a country music fan will pay more attention to taylor swift's songs.

To capture the above-mentioned intuitions in weighting items in a bundle, we design an adaptive weighted sum operation which is inspired from the attention mechanism in neural

networks [Chen *et al.*, 2017]. Let $\mathbf{b}_s$ be the embedding for bundle b_s, we obtain it by:

$$\mathbf{b}_s = \sum_{j \in \mathcal{G}_s} \frac{\alpha(i,j)}{\sum_{j' \in \mathcal{G}_s} \alpha(i,j')} \mathbf{v}_j, \tag{1}$$

where $\alpha(i,j)$ denotes weight of item v_j when user u_i considers whether to interact with a bundle, and we add L_1 normalization on the weights to eliminate the impact of varied sizes of different bundles. Directly learning $\alpha(i,j)$ from data has generalization issue, since it cannot learn values for unseen user-item pairs. As such, we need to parameterize $\alpha(i,j)$ to increase its generalization ability. According to the design of neural attention networks [Chen *et al.*, 2017], a common way is to place a multi-layer perceptron (MLP) above user embedding and item embedding to estimate $\alpha(i,j)$ as: $\alpha(i,j) = \exp(\mathbf{z}^T \sigma(\mathbf{P}_v \mathbf{v}_j + \mathbf{P}_u \mathbf{u}_i))$, where $\mathbf{u}_i$ denotes the embedding for user u_i, $\mathbf{P}_u, \mathbf{P}_v$, and $\mathbf{z}$ are parameters of the MLP, and σ is the activation function.

Factorized Attention Network

However, such MLP structure is inefficient to learn the low-rank (i.e., multiplicative) relation between user and item, as evidenced in a recent work [Beutel *et al.*, 2018]. Since the observed interactions between users and items are sparse in nature, they are likely to follow a low-rank structure. As such, we apply a low-rank model to parameterize $\alpha(i,j)$:

$$\alpha(i,j) = \exp(\mathbf{u}_i^T \mathbf{a}_j), \tag{2}$$

where $\mathbf{a}_j$ is the vector to characterize item v_j in making decision towards an item in a bundle. We call $\mathbf{a}_j$ as the *item attention vector*, which we intentionally make it different from item embedding $\mathbf{v}_j$ to increase the model's capability. This is to cover the case that a user likes a separate item, but dislikes it in a bundle, for example, some clothing products are good-looking but are hard to form a good match with others.

It is worth noting that our design of the weighting strategy can be seen as factorizing the attention weight matrix $\mathbf{A} = [\alpha(i,j)] \in \mathbb{R}^{N \times M}$ with a low-rank model followed by a softmax operation, being different from existing designs of neural attention network. As such, we term it as *factorized attention network*.

3.2 Joint Modeling of User-Bundle and User-Item Interactions

After obtaining a bundle's representation, we can easily get a user-bundle predictive model by feeding user embedding and bundle embedding into a matching function:

$$\hat{r}_{is} = f_{bundle}(\mathbf{u}_i, \mathbf{b}_s), \tag{3}$$

where f_{bundle} denotes the matching function for user-bundle prediction, and $\mathbf{b}_s$ is the bundle embedding defined in Equation (1). Many choices of matching function explored in collaborative filtering literature can be applied here, such as inner product used in matrix factorization methods [Koren, 2009] and distance functions used in metric learning methods [Hsieh *et al.*, 2017]. Here we choose the neural collaborative filtering (NCF) framework [He *et al.*, 2017] for two reasons: 1) NCF is a neural network architecture, being suitable to integrate the bundle representation learning module to build an end-to-end model; 2) NCF is more flexible in designing multiple nonlinear layers to learn complex matching function. The flexibility allows us to seamlessly incorporate user-item interaction modeling into user-bundle model.

Joint User-Bundle and User-Item Model

To transfer the information learned from more rich user-item interactions, the user embeddings and item embeddings are shared in the two tasks of user-bundle prediction and user-item prediction. As such, the user-item predictive model could be abstracted as follows:

$$\hat{h}_{ij} = f_{item}(\mathbf{u}_i, \mathbf{v}_j), \tag{4}$$

where f_{item} denotes the matching function for user-item prediction. Next we consider how to design the two matching functions by relating the two tasks.

A straightforward solution is to use MLP as the matching function [He *et al.*, 2017], and employ two separate MLP for f_{bundle} and f_{item}:

$$\begin{aligned} \hat{r}_{is} &= MLP_{bundle}(\mathbf{u}_i, \mathbf{b}_s), \\ \hat{h}_{ij} &= MLP_{item}(\mathbf{u}_i, \mathbf{v}_j), \end{aligned} \tag{5}$$

where MLP_{bundle} are MLP_{item} are the multi-layer matching network for user-bundle and user-item prediction tasks, respectively. However, this solution is insufficient to make the two tasks benefit from each other, because the parameters of the two MLP are totally separate without any connection. Considering the two tasks share the same input of user embedding $\mathbf{u}_i$, we speculate that the low layers of the two MLP may do the same thing, i.e., extracting signal about user preference based on certain item properties, such as price, brand, topic (or subcategory), among others. For this reason, we propose the joint modeling architecture as shown in Figure 2, where the low layers of the two MLP are shared and high layers are different to be task-specific. This architecture is formulated as follows:

$$\begin{aligned} \hat{r}_{is} &= MLP_{bundle}(MLP_{shared}(\mathbf{u}_i, \mathbf{b}_s)), \\ \hat{h}_{ij} &= MLP_{item}(MLP_{shared}(\mathbf{u}_i, \mathbf{v}_j)), \end{aligned} \tag{6}$$

where MLP_{shared} denotes the shared layers to extract low-level prediction signal from the inputs of the two tasks. The output of MLP_{shared} is a vector which is then fed into two separate MLP to make task-specific predictions.

Note that this architecture is inspired from [Yosinski *et al.*, 2014], which transfers low-level features of convolutional neural networks across computer vision tasks. Such an architecture, to our knowledge, is first introduced to recommendation to align two related prediction tasks. Recently, [Cao *et al.*, 2018] presents a joint model to correlate user-item and group-item prediction tasks. However, their method only has shared layers, which can be seen as a special case of our method by removing the task-specific MLP.

Multi-Task Learning

Given an observed interaction (click, purchase or review) of user u_i on bundle b_s, we can know that u_i is interested in b_s. However, for the opposite case that there is no interaction between u_i and bundle b_t, we cannot conclude the user is

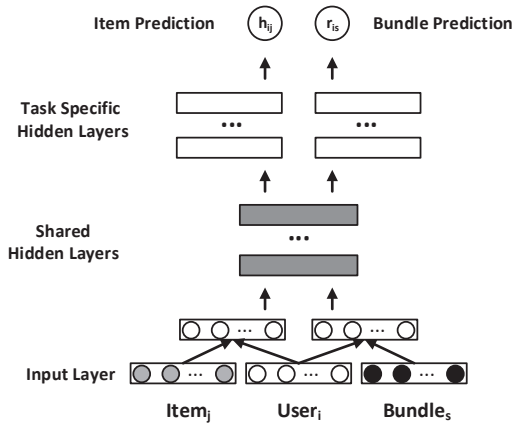

Figure 2: Illustration of joint modeling on user-item interaction and user-bundle interaction.

not interested in b_t, since it is likely that the user does not know the bundle before. As such, a more reasonable way is to assume that the chosen bundle is more preferable to the unchosen bundles for the user. This motivates the use of a pairwise learning framework, which we adopt the Bayesian Personalized Ranking (BPR) loss [Rendle *et al.*, 2009]:

$$L_{bundle} = \sum_{i=1}^{N} \sum_{s \in \mathcal{R}_i} \sum_{t \notin \mathcal{R}_i} -\ln \sigma(\hat{r}_{is} - \hat{r}_{it}) + \lambda_b ||\Theta_b||_2^2, \quad (7)$$

where $\mathcal{R}_i$ denotes all bundles interacted by user u_i. We can treat (i, s) as a positive example and (i, t) as negative, such that minimizing the loss function forces the prediction $\hat{r}_{is}$ to be larger than $\hat{r}_{it}$. $\sigma(\cdot)$ is the sigmoid function, and Θ_b is the model parameter set of bundle prediction task. L_2 regularization is applied to prevent overfitting.

Given the similar motivation of learning from user-item interactions, we define the objective function for the user-item prediction task as follows:

$$L_{item} = \sum_{i=1}^{N} \sum_{j \in \mathcal{H}_i} \sum_{l \notin \mathcal{H}_i} -\ln \sigma(\hat{h}_{ij} - \hat{h}_{il}) + \lambda_i ||\Theta_i||_2^2, \quad (8)$$

where $\mathcal{H}_i$ denotes all items interacted by user u_i. Similarly, we can treat (i, j) as a positive example and (i, l) as a negative example for the user-item prediction task. Θ_i is the model parameter set of item prediction task.

The final objective function for the joint model is the sum of the objective functions of the two prediction tasks:

$$L = L_{bundle} + L_{item}. \quad (9)$$

In each iteration, we first sample a user u_i. We then sample the positive example (i, s) with a paired negative sample (i, t), as well as the positive example (i, j) with a paired negative sample (i, l). Two gradient steps are then performed to minimize L_{bundle} and L_{item} sequentially. The above steps are iteratively executed until convergence (which is monitored by the accuracy change of the user-bundle recommendation task in the validation set). Note that this stochastic learning algorithm can be easily extended to mini-batch training by sampling multiple users in a batch.

Dataset	NetEase	Youshu
# User	18,528	8,039
# Bundle	22,864	4,771
# Item	123,628	32,770
# User-Bundle	302,303	51,377
# User-Item	1,128,065	138,515
# Bundle-Item	1,778,838	176,667
# Avg. Bundle interactions	16.32	6.39
# Avg. Item interactions	60.88	17.23
# Avg. Bundle size	77.80	37.03
# User-Item density	0.05%	0.05%
# User-Bundle density	0.07%	0.13%

Table 1: Dataset Statistics

4 Experiments

In this section, we conduct experiments on two real-world datasets aiming to answer following research questions:

RQ1 Can our proposed DAM outperform the state-of-the-art bundle recommendation models?

RQ2 How is the effectiveness of factorized attention network? Can it perform better than other aggregation strategies?

RQ3 Can multi-task learning framework improve DAM's performance? How does the number of shared layer affect model performance?

4.1 Experimental Settings

Dataset. Our proposed model is evaluated on two datasets. The first dataset is provided by the work of EFM [Cao *et al.*, 2017]. The authors crawled data from the Netease Cloud Music[1], which enables users to construct a list of songs with a specific theme. Each bundle consists of at least 5 items, i.e., songs, each item appears in at least 5 bundles, and users listen to at least 10 items and 10 bundles. The other dataset is constructed by crawling data from Youshu, a Chinese book review site[2]. Similar to NetEase, a user can construct a list of books they desired. The statistics of datasets are shown in Table 1.

Evaluation Metric. We employ *leave-one-out* evaluation protocol. For each user, one of her interactions are randomly removed for testing. To evaluate the top-K recommendation performance of models, we employ a relevance-based metric – *Recall@K* that measures the number of positive items presenting within the top-K recommendation list and a ranking-based metric – *MAP@K* considering the ranking positions of the positive items within the top-K recommendation list. Since ranking all the bundles for users is too time-consuming, we randomly select 99 bundles that are not interacted with users as negative samples.

Baselines. To show the effectiveness of DAM, our method is compared to the following models:

[1]https://music.163.com/

[2]http://www.yousuu.com/

Model	Netease				Youshu			
	d=5		d=10		d=5		d=10	
	Recall@5	MAP@5	Recall@5	MAP@5	Recall@5	MAP@5	Recall@5	MAP@5
BPR	0.3392	0.1967	0.3389	0.2018	0.5274	0.3523	0.5362	0.3608
NCF	0.3482	0.2005	0.3534	0.2076	0.5627	0.3723	0.5715	0.3797
BR	0.3241	0.1951	0.3318	0.1966	0.5669	0.3746	0.5752	0.3748
EFM	**0.3512**	**0.2039**	**0.3654**	**0.2140**	**0.5844**	**0.3863**	**0.5930**	**0.3903**
DAM	**0.3882**	**0.2321**	**0.4016**	**0.2412**	**0.5967**	**0.4025**	**0.6109**	**0.4121**
Improv.	+10.54%	+13.83%	+9.91%	+12.71%	+2.10%	+4.19%	+3.02%	+5.59%

Table 2: Top-K recommendation performance comparison of different methods. The last row Improv. denotes the relative improvement over the baseline:DAM outperforms all baselines on all metrics.

- **BPR** [Rendle *et al.*, 2009]: BPR is the basic pairwise ranking algorithm based on implicit feedback. We optimize the BPR ranking loss under the matrix factorization framework.
- **NCF** [He *et al.*, 2017]: This method is a state-of-the-art neural CF model which combines element-wise and hidden layers of the concatenation of user and item embedding to capture their high-order interactions.
- **BR** [Pathak *et al.*, 2017]: BR is a two-step model, where the latent vectors of users and items are learned in the first step, and item latent vectors are aggregated to predict the user preference of bundles.
- **EFM** [Cao *et al.*, 2017]: EFM extend the traditional factorization model by incorporating the item-item-bundle co-occurrence information. It jointly factorizes the user-item-bundle interactions matrix and item-item-bundle co-occurrence sparse shifted positive pointwise mutual information matrix by optimizing the BPR ranking loss.

Hyper-parameter settings. We implement all the models based on Pytorch. For these methods, the learning rate is searched in [0.01, 0.005, 0.001, 0.0005]. The training batch size is fixed to 1024, and Adam algorithm is utilized to optimize all the baselines. The coefficient of L_2 regularization is ranged in [0.1, 0.01, 0.001, 0.0001]. The layer number of NCF is set to 2 where the embedding size of each layer is the same. We use dropout to prevent NCF and DAM from overfitting where the dropout ratio is searched ranging from 0.1 to 0.9. without specification, we use three-layer structure for DAM with two shared layer. The embedding size of each layer remains the same.

4.2 Performance Comparison (RQ1)

Top-K performance comparison of DAM and other state-of-the-art models under different embedding size are reported in Table 2. BPR achieves poor performance in both datasets, indicating that directly optimize user and bundle embedding through inner product is insufficient to capture the user preference towards bundles. These results verify that the insufficient interaction data between user and bundle limits the model performance. Compared to BPR, NCF achieves better performance, demonstrating the effectiveness of applying deep neural network to learn the nonlinear user-bundle feature interactions. By comparing BR with EFM and DAM, we can see that co-training the user-item and user-bundle interactions perform better, demonstrating that jointly utilizing these two types of interaction data for training is beneficial to model performance.

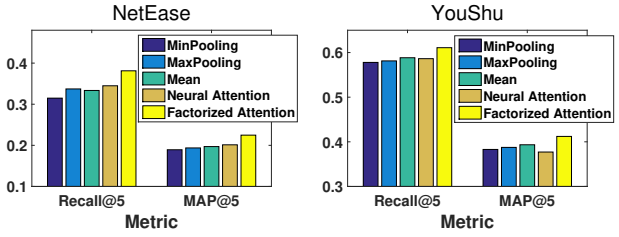

Figure 3: The performance comparison of different fusing strategies.

DAM achieves the best performance under all circumstances. In particular, DAM achieves 9.91%, 12.71% and 3.02%, 5.59% performance improvement against the strongest baseline w.r.t R@5 and MAP@5 when the embedding size is 10, demonstrating the effectiveness of our model. The improvement could be attributed to the attentive modeling of item weights and the multi-task learning framework.

4.3 Effect of Factorized Attention (RQ2)

Factorized attention network is an important component of DAM. Therefore, to investigate the effectiveness of factorized attention network, we compare it with other fusing strategies, including min pooling, max pooling, mean pooling and neural attention. Figure 3 shows the performance comparison of different fusing strategy. MinPooling, maxPooling, meanPooling and neural attention achieve similar performance, indicating these methods are insufficient to capture the complex relations among the items. Factorized attention achieves the best performance in all cases, demonstrating the effectiveness of capturing the user attention by the low-rank structure. Although both neural attention and factorized attention learn dynamic weights for users, factorized attention achieves a better performance, demonstrating that factorized attention has better generalization performance than neural attention.

4.4 Effect of Shared Layer Number (RQ3)

To investigate whether multi-task learning improves the model performance and how does it affect the model performance, we vary the number of shared layers. We use DAM-k to denote DAM with k shared layers while the hidden layer number is fixed to 3. DAM-0 could be viewed as DAM without utilizing the multi-task learning framework. The exper-

Model	NetEase		YouShu	
	Recall@5	MAP@5	Recall@5	MAP@5
DAM-0	0.3812	0.2286	0.5842	0.3980
DAM-1	0.3999	0.2397	0.6007	0.4071
DAM-2	0.4016	0.2412	**0.6109**	**0.4121**
DAM-3	**0.4048**	**0.2427**	0.6072	0.4111

Table 3: Top-K recommendation performance comparison w.r.t the shared layer number. DAM-k represents the DAM model with k shared layers.

imental results are shown in Table 3. By comparing DAM-1, DAM-2 and DAM-3 with DAM-0, we can see that multi-task learning models perform better than single-task learning model, demonstrating that jointly utilizing the user-item and user-bundle interactions enhance the model performance. As we can see from the table, increasing the number of shared layers enhances the recommendation performance. DAM-2 achieves performance improvement over DAM-1 in both datasets, demonstrating applying more shared layers is beneficial to the model performance. The improvement could be attributed to the better modeling of user preference over items. When further increasing the number of shared layers, DAM-3 only achieves better performance on the NetEase dataset. The reason is that applying too many shared layers might introduce noises irrelevant to the bundle recommendation task.

5 Related Work

5.1 Bundle Recommendation

In recommendation domain, several efforts [Garfinkel *et al.*, 2006] have been made in solving the bundle recommendation problem. Sar Shalom *et al.* [Sar Shalom *et al.*, 2016] aimed to recommend a list of items to users by optimizing the list's click-through rate. Liu *et al.* [Liu *et al.*, 2017] estimated the probability that a consumer would buy an item together with the one already brought. Probabilistic models have been used to maximize the expected revenue of a recommendation bundle [Beladev *et al.*, 2016] and incorporating the factors such as user impact, package viability and recommendation fairness [Qi *et al.*, 2016]. Moreover, with the development of pairwise ranking approach, some works tried to rank the bundle by optimizing a pairwise ranking loss. Embedding Factorization Machine (EFM) [Cao *et al.*, 2017] and List Recommendation Model (LIRE) [Liu *et al.*, 2014] simultaneously considered the users' previous interactions with both individual items and lists under BPR framework. The bundle BPR Model [Pathak *et al.*, 2017] made use of the parameters previously learned through an item BPR model.

5.2 Deep Learning for Recommendation

Deep learning methods [He *et al.*, 2017; Xue *et al.*, 2017; Wang *et al.*, 2019] have been widely used to improve the recommendation performance. The pioneer work Neural Collaborative Filtering (NCF) [He *et al.*, 2017] replaced the inner product with a neural architecture to learn interactions between users and items. Deep learning on graph for recommendation is one of trending topics [Wang *et al.*, 2019]. The closest work of deep learning to ours was done by Yosinski *et al.* [Yosinski *et al.*, 2014]. They demonstrated an approach for quantifying the transferability of features from each layer of a neural network.

Multiple works [Chen *et al.*, 2017; Cao *et al.*, 2018; Chen *et al.*, 2018] incorporated the attention mechanism with the implementation of neural networks proposed to improve the performance of recommender systems. Attentive Collaborative Filtering(ACF) [Chen *et al.*, 2017] aggregated the item- and component-level implicit feedback with an attention network to get the representation for a multimedia item. The AGREE model [Cao *et al.*, 2018] learned to assign an attention weight for members to solve group recommendation problem.

Our work is orthogonal to the above mentioned work, as we exploit the deep neural network to tackle the bundle recommendation task under the multi-task learning framework. Moreover, we employ the attention mechanism to learn the representation of bundle as well. However, instead of using the neural attention network, we design a weighting strategy which can be seen as factorizing the attention weight matrix with a low-rank model followed by a softmax operation.

6 Conclusion and Future Work

In this paper, we present a novel deep attentive multi-task model for bundle recommendation. We first propose a factorized attention network to aggregate the item information of each bundle. Experimental results demonstrate that our attention mechanism can effectively learn a personalized weight for each user. To utilize user-item and user-bundle information simultaneously, we propose a multi-task neural network to share the knowledge of two tasks (e.g., user-item modeling and user-bundle modeling). We analyze the effect of each layer and find that sharing the representation of the first three layers could enhance the recommendation performance. The overall comparison with start-of-the-art methods demonstrate the effectiveness of our method.

In future, we plan to extend our work in the following directions: (1) Modeling the item co-occurrence information. In e-commerce, some products may have a complementary relationship, e.g., tennis racket and tennis, eyeglass and glasses frame. (2) Realizing the bundle recommendation with temporal dynamics [Koren, 2009]. Users' interests evolve over time. In some real-world scenarios such as music listening, it is worthwhile to generate dynamic bundle recommendation by considering the time factor.

Acknowledgements

The paper was supported by the National Key Research and Development Program (2017YFB0202201), the National Natural Science Foundation of China (61702568, U1711267), the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (2017ZT07X355) and the Fundamental Research Funds for the Central Universities under Grant (17lgpy117).

References

[Beladev *et al.*, 2016] Moran Beladev, Lior Rokach, and Bracha Shapira. Recommender systems for product bundling. *Knowledge-Based Systems*, 111:193–206, 2016.

[Beutel *et al.*, 2018] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H. Chi. Latent cross: Making use of context in recurrent recommender systems. In *WSDM*, pages 46–54, 2018.

[Cao *et al.*, 2017] Da Cao, Liqiang Nie, Xiangnan He, Xiaochi Wei, Shunzhi Zhu, and Tat-Seng Chua. Embedding factorization models for jointly recommending items and user generated lists. In *SIGIR*, pages 585–594, 2017.

[Cao *et al.*, 2018] Da Cao, Xiangnan He, Lianhai Miao, Yahui An, Chao Yang, and Richang Hong. Attentive group recommendation. In *SIGIR*, pages 645–654, 2018.

[Chen *et al.*, 2017] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *SIGIR*, pages 335–344, 2017.

[Chen *et al.*, 2018] Liang Chen, Yang Liu, Zibin Zheng, and Philip S. Yu. Heterogeneous neural attentive factorization machine for rating prediction. In *CIKM*, pages 833–842, 2018.

[Cheng *et al.*, 2016] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *DLRS*, pages 7–10, 2016.

[Garfinkel *et al.*, 2006] Robert Garfinkel, Ram Gopal, Arvind Tripathi, and Fang Yin. Design of a shopbot and recommender system for bundle purchases. *Decision Support Systems*, 42(3):1974–1986, 2006.

[Ge *et al.*, 2014] Yong Ge, Hui Xiong, Alexander Tuzhilin, and Qi Liu. Cost-aware collaborative filtering for travel tour recommendations. *TOIS*, 32(1):4, 2014.

[He and Chua, 2017] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *SIGIR*, pages 355–364, 2017.

[He *et al.*, 2017] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, pages 173–182, 2017.

[Hsieh *et al.*, 2017] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. Collaborative metric learning. In *WWW*, pages 193–201, 2017.

[Koren, 2009] Yehuda Koren. Collaborative filtering with temporal dynamics. In *KDD*, pages 447–456, 2009.

[Liu *et al.*, 2014] Yidan Liu, Min Xie, and Laks VS Lakshmanan. Recommending user generated item lists. In *RecSys*, pages 185–192, 2014.

[Liu *et al.*, 2017] Guannan Liu, Yanjie Fu, Guoqing Chen, Hui Xiong, and Can Chen. Modeling buying motives for personalized product bundle recommendation. *TKDD*, 11(3):28, 2017.

[Pathak *et al.*, 2017] Apurva Pathak, Kshitiz Gupta, and Julian McAuley. Generating and personalizing bundle recommendations on steam. In *SIGIR*, pages 1073–1076, 2017.

[Qi *et al.*, 2016] Shuyao Qi, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. Recommending packages to groups. In *ICDM*, pages 449–458, 2016.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[Sar Shalom *et al.*, 2016] Oren Sar Shalom, Noam Koenigstein, Ulrich Paquet, and Hastagiri P Vanchinathan. Beyond collaborative filtering: The list recommendation problem. In *WWW*, pages 63–72, 2016.

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

[Wang *et al.*, 2018a] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. Tem: Tree-enhanced embedding model for explainable recommendation. In *WWW*, pages 1543–1552, 2018.

[Wang *et al.*, 2018b] Zihan Wang, Ziheng Jiang, Zhaochun Ren, Jiliang Tang, and Dawei Yin. A path-constrained framework for discriminating substitutable and complementary products in e-commerce. In *WSDM*, pages 619–627, 2018.

[Wang *et al.*, 2019] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. *SIGIR*, 2019.

[Xie *et al.*, 2018a] Fenfang Xie, Liang Chen, Yongjian Ye, Yang Liu, Zibin Zheng, and Xiaola Lin. A weighted metagraph based approach for mobile application recommendation on heterogeneous information networks. In *ICSOC*, pages 404–420, 2018.

[Xie *et al.*, 2018b] Fenfang Xie, Liang Chen, Yongjian Ye, Zibin Zheng, and Xiaola Lin. Factorization machine based service recommendation on heterogeneous information networks. In *ICWS*, pages 115–122, 2018.

[Xu *et al.*, 2018] Jun Xu, Xiangnan He, and Hang Li. Deep learning for matching in search and recommendation. In *SIGIR*, pages 1365–1368, 2018.

[Xue *et al.*, 2017] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. Deep matrix factorization models for recommender systems. In *IJCAI*, pages 3203–3209, 2017.

[Yosinski *et al.*, 2014] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *NIPS*, pages 3320–3328, 2014.

[Zhu *et al.*, 2014] Tao Zhu, Patrick Harrington, Junjun Li, and Lei Tang. Bundle recommendation in ecommerce. In *SIGIR*, pages 657–666, 2014.