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

Liang Chen¹,², Yang Liu¹,², Xiangnan He³, Lianli Gao⁴ and Zibin Zheng¹,²∗
¹School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China
²Guangdong Key Laboratory for Big Data Analysis and Simulation of Public Opinion, The school of Communication and Design, Sun Yat-sen University, Guangzhou, China
³University of Science and Technology of China
⁴The 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].

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

![Figure 1: Example of product bundles (relevant songs and complementary electronic products).](image-url)
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 \( \mathbf{R} = [r_{js}]_{N \times K} \) 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:

\[
\mathbf{b}_s = \sum_{j \in G_s} \frac{\alpha(i, j)}{\sum_{j' \in G_s} \alpha(i, j')} \mathbf{v}_j,
\]

(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 useful 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:

\[
\alpha(i, j) = \exp (\mathbf{z}^T \sigma(\mathbf{P}_u v_j + \mathbf{P}_v u_i)),
\]

(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 intentionally make it different from item embedding \( v_j \) 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:

\[
\hat{r}_{is} = f_{\text{bundle}}(u_i, b_s),
\]

(3)

where \( f_{\text{bundle}} \) denotes the matching function for user-bundle prediction, and \( b_s \) 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:

\[
\hat{h}_{ij} = f_{\text{item}}(u_i, v_j),
\]

(4)

where \( f_{\text{item}} \) 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 \( f_{\text{bundle}} \) and \( f_{\text{item}} \):

\[
\hat{r}_{is} = MLP_{\text{bundle}}(u_i, b_s),
\]

\[
\hat{h}_{ij} = MLP_{\text{item}}(u_i, v_j),
\]

(5)

where \( MLP_{\text{bundle}} \) are \( MLP_{\text{item}} \) 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 \( u_i \), 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:

\[
\hat{r}_{is} = MLP_{\text{bundle}}(MLP_{\text{shared}}(u_i, b_s)),
\]

\[
\hat{h}_{ij} = MLP_{\text{item}}(MLP_{\text{shared}}(u_i, v_j)),
\]

(6)

where \( MLP_{\text{shared}} \) denotes the shared layers to extract low-level prediction signal from the inputs of the two tasks. The output of \( MLP_{\text{shared}} \) 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 \( b_s \), we cannot conclude the user is...
In each iteration, we first sample a user \( u \) prediction task as follows: interactions, we define the objective function for the user-item prediction task as a positive example \((i,s)\) as a negative example. 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 \( \Theta_b \) is the model parameter set of item 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:

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

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. \( \Theta_i \) 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:

\[
L = L_{bundle} + L_{item}. \tag{9}
\]

In each iteration, we first sample a user \( u \). 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.

**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 present 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/

---

**Table 1: Dataset Statistics**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>NetEase</th>
<th>Youshu</th>
</tr>
</thead>
<tbody>
<tr>
<td># User</td>
<td>18,528</td>
<td>4,771</td>
</tr>
<tr>
<td># Bundle</td>
<td>22,864</td>
<td>7,780</td>
</tr>
<tr>
<td># Item</td>
<td>123,628</td>
<td>37,037</td>
</tr>
<tr>
<td># Bundle</td>
<td>1,128,065</td>
<td>176,667</td>
</tr>
<tr>
<td># Avg. Bundle interactions</td>
<td>16.32</td>
<td>27.80</td>
</tr>
<tr>
<td># Avg. Item interactions</td>
<td>60.88</td>
<td>17.23</td>
</tr>
<tr>
<td># Avg. Bundle size</td>
<td>77.80</td>
<td>30.23</td>
</tr>
<tr>
<td># User-Bundle density</td>
<td>0.05%</td>
<td>0.05%</td>
</tr>
<tr>
<td># User-Item density</td>
<td>0.07%</td>
<td>0.13%</td>
</tr>
</tbody>
</table>

---

**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?
### 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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Metric</th>
<th>Netease d=5</th>
<th>Netease d=10</th>
<th>Youshu d=5</th>
<th>Youshu d=10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Recall@5</td>
<td>MAP@5</td>
<td>Recall@5</td>
<td>MAP@5</td>
<td>Recall@5</td>
</tr>
<tr>
<td>BPR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cao et al., 2017</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NCF</td>
<td>0.3392</td>
<td>0.1967</td>
<td>0.3389</td>
<td>0.2018</td>
<td>0.5274</td>
</tr>
<tr>
<td>He et al., 2017</td>
<td>0.3482</td>
<td>0.2005</td>
<td>0.3534</td>
<td>0.2076</td>
<td>0.5627</td>
</tr>
<tr>
<td>BR</td>
<td>0.3241</td>
<td>0.1951</td>
<td>0.3318</td>
<td>0.1966</td>
<td>0.5669</td>
</tr>
<tr>
<td>EFM</td>
<td>0.3512</td>
<td>0.2039</td>
<td>0.3654</td>
<td>0.2140</td>
<td>0.5844</td>
</tr>
<tr>
<td>DAM</td>
<td>0.3882</td>
<td>0.2321</td>
<td>0.4016</td>
<td>0.2412</td>
<td>0.5967</td>
</tr>
<tr>
<td>Improv.</td>
<td>+10.54%</td>
<td>+13.83%</td>
<td>+9.91%</td>
<td>+12.71%</td>
<td>+2.10%</td>
</tr>
</tbody>
</table>

### Figure 3: The performance comparison of different fusing strategies.

- **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.

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-
Experimental 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.

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

<table>
<thead>
<tr>
<th>Model</th>
<th>NetEase</th>
<th>YouShu</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Recall@5</td>
<td>MAP@5</td>
</tr>
<tr>
<td>DAM-0</td>
<td>0.3812</td>
<td>0.2286</td>
</tr>
<tr>
<td>DAM-1</td>
<td>0.3999</td>
<td>0.2397</td>
</tr>
<tr>
<td>DAM-2</td>
<td>0.4016</td>
<td>0.2412</td>
</tr>
<tr>
<td>DAM-3</td>
<td>0.4048</td>
<td>0.2427</td>
</tr>
</tbody>
</table>

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, eyeglasses 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


