Personalized Bundle Recommendation in Online Games

Qilin Deng
NetEase Games
Hangzhou, Zhejiang, China
dengqilin@corp.netease.com

Kai Wang
NetEase Games
Hangzhou, Zhejiang, China
wangkai02@corp.netease.com

Minghao Zhao
NetEase Games
Hangzhou, Zhejiang, China
zhaominghao@corp.netease.com

Zhene Zou
NetEase Games
Hangzhou, Zhejiang, China
zouzhene@corp.netease.com

Runze Wu
NetEase Games
Hangzhou, Zhejiang, China
wurunze1@corp.netease.com

Jianrong Tao
NetEase Games
Hangzhou, Zhejiang, China
hztaojianrong@corp.netease.com

Changjie Fan
NetEase Games
Hangzhou, Zhejiang, China
fanchangjie@corp.netease.com

Liang Chen
Sun Yat-Sen University
Guangzhou, Guangdong, China
chenliang6@mail.sysu.edu.cn

ABSTRACT

In business domains, *bundling* is one of the most important marketing strategies to conduct product promotions, which is commonly used in online e-commerce and offline retailers. Existing recommender systems mostly focus on recommending individual items that users may be interested in. In this paper, we target at a practical but less explored recommendation problem named bundle recommendation, which aims to offer a combination of items to users. To tackle this specific recommendation problem in the context of the *virtual mall* in online games, we formalize it as a link prediction problem on a user-item-bundle tripartite graph constructed from the historical interactions, and solve it with a neural network model that can learn directly on the graph-structure data. Extensive experiments on three public datasets and one industrial game dataset demonstrate the effectiveness of the proposed method. Further, the bundle recommendation model has been deployed in production for more than one year in a popular online game developed by Netease Games, and the launch of the model yields more than 60% improvement on conversion rate of bundles, and a relative improvement of more than 15% on gross merchandise volume (GMV).

CCS CONCEPTS

• **Information systems** → **Learning to rank**; **Recommender systems**.

KEYWORDS

recommender system; bundle recommendation; neural networks; deep learning; graph neural networks; link prediction

1 INTRODUCTION

Recommender system, which is an effective tool to alleviate the information overload, is widely used in modern e-commerce websites and online service business, e.g., Amazon, Taobao, Netflix. The basic goal of a recommender system is to find potentially interesting items for a user. Existing recommender systems mostly focus on recommending individual items to users, such as the extensive efforts on collaborative filtering that directly models the interaction between users and items.

Figure 1: The Bundle GUI in the Game *Love is Justice*.

In addition to consuming items individually, bundles are also ubiquitous in real-world scenarios. A bundle is a collection of items (products or services) consumed as a whole, and it usually reflects the frequent items which are appealing to most customers. In traditional business domains, e.g., supermarkets and offline retailers, it often takes bundling as a critical marketing strategy to attract

customers and increase sales revenue. Moreover, the combination of items are especially ubiquitous on online service platforms, e.g., the music playlist on *Spotify*, the book lists on *GoodReads*, the boards of pins on *Pinterest*, and the game bundles on *Steam*.

In Figure 1, we show the bundle recommendation scenario in the online game *Love is Justice*[1], a popular mobile game developed by *Netease Games*[2], where users could impersonate specific roles and experience beautiful antique scenes and romantic plots, and their purchase and behavior data are extensively tracked by the game server. Here, we give some brief introductions to the Graphical User Interface (GUI) involved in this study. Once the user enters the homepage of the game interface (shown in the figure), the system will occasionally pop up a personalized discount bundle to attract the user, or the user himself can check the discount bundle when he wants. In addition, the items included in the bundle can also be purchased separately, although they are not discounted. According to our analysis of purchase statistics, more than **65%** of game revenue comes from these discounted bundles, which also shows that it is profitable to increase the conversion rate of these personalized bundles.

In this paper, we address the problem of bundle recommendation in the context of online games, which aims to provide game players with the pre-defined bundles (combination of items) they are most likely to be interested in. Intuitively, this particular recommendation problem can be solved by treating bundles as "items" and then using traditional recommendation algorithms such as collaborative filtering. However, such straightforward solutions do not work well to capture user preference over bundles due to the following three difficulties:

- **Data sparsity and cold-start.** Compared with user-item interactions, user-bundle interactions are usually more sparse due to the exponential combination characteristics of bundles and limited exposure resources. And only if the user is satisfied with the item combination or the discounted price is attractive, the user will have a strong willingness to buy the bundles rather than individual items, which makes the user-bundle interaction data appear more sparse.
- **Generalization over bundles.** Previous item recommendation algorithms may rely on item-level content features (e.g., category and brand in e-commerce), and user-item collaborative relationships. However, there is usually no informative bundle-level content features in the bundle recommendation scenario. This makes it difficult to provide the model's generalization ability in bundle preference prediction through a content-based model.
- **Correlation within the bundles.** The items within the bundle are usually highly correlated and compatible. Compared to typical item recommendation, the bundle recommendation problem is more complex considering that the user-bundle preference is a nontrivial combination of user-item preference. And directly modeling the interaction effect between items remains largely unexplored in the field of recommender systems.
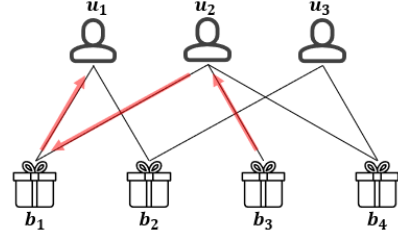
Figure 2: A toy example of a user-bundle bipartite graph with edges representing observed user-bundle interactions. The red arrow lines denote message passing paths.

Building on recent progress in deep learning on graph-structured data, we introduce a learning framework based on differentiable message passing on the user-item-bundle tripartite interaction graph constructed from historical data, and formalize the bundle recommendation problem as the link prediction problem in the tripartite graph. To alleviate the sparsity of user interactions on bundles, we integrate user-items interactions that provide additional information on user interests. To account for the compositional similarity between bundles, we derive the bundle representation by aggregating the item representations, which provides a natural good generalization ability over different bundles. We also model the correlation between bundle items in the form of learnable transformation parameters. Finally, we unify these improvements in our proposed framework named *BundleNet*, and the multi-layer message passing structure can capture the *high-order* and *multi-path* interactions over the user-item-bundle tripartite graph. As shown in Figure 2, bundle b_3 can reach user u_1 through the path $b_3 \rightarrow u_2 \rightarrow b_1 \rightarrow u_1$, and similar for bundle b_4. Moreover, compared with b_3, b_4 is a more reliable recommendation for u_1, since intuitively there is only one path existing between u_1 and b_3, while two paths connecting u_1 to b_4. Overall, the main contributions of this paper are summarized as follows:

- We explore the promising yet challenging problem of bundle recommendation in the context of online games, and provide a practical case for the application of deep learning methods in the industry.
- We employ a differentiable message passing framework to effectively capture the user preferences for bundles, which can incorporate the intermediate role of items between users and bundles on the user-item-bundle tripartite graph.
- Extensive offline experiments on both in-game and other real-world datasets are conducted to verify the effectiveness of the proposed model. Further, we deploy the whole framework online and demonstrate its effective performance through online A/B Testing.

2 PROBLEM DEFINITION

Suppose we have users $\mathcal{U} = \{u_i | i = 1, 2, ..., N_u\}$, items $\mathcal{I} \in \{i_j | j = 1, 2, ..., N_i\}$, and bundles $\mathcal{B} \in \{b_k | k = 1, 2, ..., N_b\}$, where the size of these sets is $|\mathcal{U}| = N_u, |\mathcal{I}| = N_i, |\mathcal{B}| = N_u$ respectively, and $N = N_u + N_i + N_b$. We also have the following three interaction graphs:
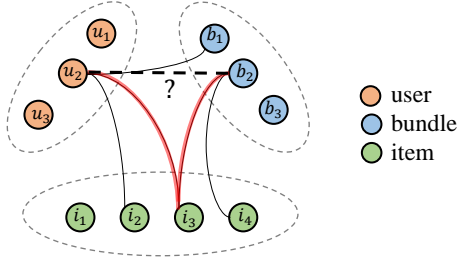
Figure 3: Problem Definition on the Tripartite Graph.

- **User-Bundle Interaction**. A user can have an interaction (e.g., click, purchase) on an bundle, which is represented as a binary variable in the adjacency matrix $\mathbf{A}_{ub}$ of the user-bundle bipartite graph, i.e., 1 means the existence of the interaction and 0 otherwise.
- **User-Item Interaction**. A user can also have an interaction (e.g., click, purchase) on a item, which is also represented as a binary variable in the adjacency matrix $\mathbf{A}_{ui}$ of the user-item bipartite graph, i.e., 1 means the existence of the interaction and 0 otherwise.
- **Bundle-Item Interaction**. Each bundle b is usually made up of several items, denoted as $b = \{i_1, i_2, ,, i_s\}$, where s denotes the bundle size (larger than 1), and each item i_i in the bundle belongs to the item set $\mathcal{I}$. We consider this containment relationship between bundles and their constituent items as the interaction relationship, just like the interaction between users and items. Similarly, the interaction is represented as a binary variable in the adjacency matrix $\mathbf{A}_{bi}$ of the bundle-item bipartite graph, i.e., 1 means the existence of the interaction and 0 otherwise.

In the three interaction graphs discussed above, we actually have three bipartite graphs since there are only two types of entities in each interaction graph. Based on these bipartite graphs, in this paper, we introduce a user-item-bundle tripartite graph $\mathcal{G} = (\mathcal{U}, \mathcal{I}, \mathcal{B}, \mathcal{E})$, where $\mathcal{U}$ denotes user nodes, $\mathcal{I}$ denotes item nodes, $\mathcal{B}$ denotes bundle nodes and $\mathcal{E}$ is the corresponding link edges between these nodes, including user-item, user-bundle, bundle-item interactions, as illustrated in Figure 3.

The benefits that the introduction of item interaction information can bring us are twofold. On the one hand, a user's preference for a bundle could be inferred to a certain extent by his preference for items within the bundle, i.e., the preference over two bundles that share some items may be similar. On the other hand, the bundle contains the co-occurrence relationship between items, as items within a bundle are usually gathered based on a specific theme. If the item co-occurrence signal within the bundles can be properly utilized, we may learn a better recommendation model for individual items. These mutual correlations allow the performance of user-bundle and user-item recommendations to be mutually reinforced.

Based on the constructed tripartite graph $\mathcal{G}$, we define the bundle recommendation problem as a link prediction problem on graph $\mathcal{G}$. Essentially, this problem estimates the likelihood of an edge between a user node u and a bundle node b (e.g., the node u_2 and

the node b_2 in Figure 3), which represents how likely the user will be interested in the bundle. Formally, given the interaction graph $\mathcal{G}$, we propose a neural recommendation optimization model to learn an approximation function map f as follows:

$$\hat{p} = f(u, b | \mathcal{G}; \theta) \tag{1}$$

Here, θ is the parameters of the neural model to be learned and $\hat{p}$ is the predicted likelihood that the user u matches the bundle b, which will be specified in the following subsections.

3 METHODOLOGY

We give the formal definition of the bundle recommendation problem above, in this section, we introduce the various components of the proposed model BundleNet in detail. The overall model framework is shown in Figure 4.

3.1 Embedding Layer

Following existing research work [4, 6, 17], for a tripartite graph $\mathcal{G} = (\mathcal{U}, \mathcal{I}, \mathcal{B}, \mathcal{E})$, we define $\mathbf{e}_u \in \mathbb{R}^d$, $\mathbf{e}_i \in \mathbb{R}^d$ and $\mathbf{e}_b \in \mathbb{R}^d$ as the embedding vectors of user node u, item node i and bundle node b respectively, where d is the embedding size. It can be expressed as:

$$\mathbf{e}_u = \text{EMBED}(u), \ \mathbf{e}_i = \text{EMBED}(i), \ \mathbf{e}_b = \text{EMBED}(b) \tag{2}$$

Suppose we denote the one-hot feature vector for user u as $x_u \in \mathbb{R}^N$, denote the embedding matrix of users as $\mathbf{E}_u \in \mathbb{R}^{N \times d}$, then we can obtain the user embedding vector of u by $e_u = \mathbf{E}_u^T x_u$. Likewise, we can get the embedding representation of item nodes $\mathbf{e}_u$, bundle nodes $\mathbf{e}_b$, which is omitted here. We stack these node embeddings as the input representation for subsequent modules:

$$\mathbf{E} = [\mathbf{E}_u, \mathbf{E}_i, \mathbf{E}_b] \tag{3}$$

3.2 Graph Propagation Layer

Inspired by recent convolutional neural networks that operate directly on graph-structured data, we use *Graph Convolutional Networks* (GCNs) [9] to process the tripartite graph data. GCN generalizes convolutions to graphs, which can naturally integrate both node attributes and topological structure in graphs, have been proved to be effective in representation learning for graph-structured data. Its propagation rule can be formulated as $\mathbf{Z} = f(\mathbf{X}, \mathbf{A})$, where $\mathbf{X}$ denotes node feature matrix (node embedding in this work), $\mathbf{A}$ denotes adjacency matrix of the underlying graph structure, and $\mathbf{Z}$ denotes the encoded node representation. The single-layer propagation rule is:

$$\mathbf{Z} = f(\mathbf{X}, \mathbf{A}) = \sigma(\hat{\mathbf{A}} \mathbf{X} \mathbf{W}) \tag{4}$$

Here, $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1/2}$, with $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, and $\hat{\mathbf{A}}$ can be calculated in a pre-processing step to speed up the model training. The σ denotes an element-wise activation function such as the ReLU$(\cdot) = max(0, \cdot)$. In our case, the adjacency matrix of the user-item-bundle tripartite graph is constructed as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{A}_{ui} & \mathbf{A}_{ub} \\ \mathbf{A}_{ui}^T & \mathbf{0} & \mathbf{A}_{bi}^T \\ \mathbf{A}_{ub}^T & \mathbf{A}_{bi} & \mathbf{0} \end{bmatrix} \tag{5}$$
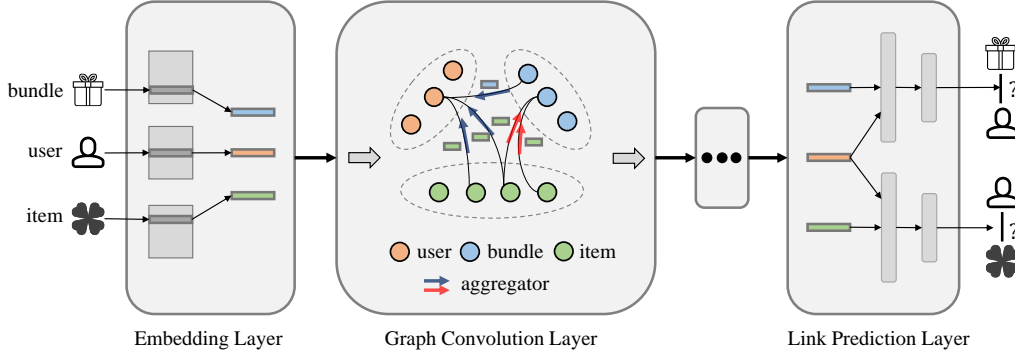
Figure 4: An illustration of the overall model framework of BundleNet.

where $\mathbf{A}_{ui}$, $\mathbf{A}_{ub}$ and $\mathbf{A}_{bi}$ denote the adjacency matrices of user-item, user-bundle and bundle-item interaction graph, respectively. Note that the elements on the main diagonal are all 0, since there is no self-loop connection edge. We can stack several layers to learn better hidden representations (high-order interactions) for graph nodes, with the following layer-wise propagation rule:

$$\mathbf{H}^{l+1} = \sigma(\hat{\mathbf{A}}\mathbf{H}^l\mathbf{W}^l) \qquad (6)$$

where, $\mathbf{H}^l$ denotes input represeatation of graph nodes in the lth layer, $\mathbf{H}^0 = \mathbf{E}$ is the embedding matrix given by formula 3, $\mathbf{W}^l$ denotes a layer-specific trainable weight matrix, $\hat{\mathbf{A}}$ is defined as above, σ denotes an element-wise activation function such as the $\text{ReLU}(\cdot) = max(0, \cdot)$ and $\mathbf{H}^{l+1}$ is the output representation matrix in the $(l+1)^{th}$ layer.

The standard GCN model is widely used in *homogeneous* graph, however, the tripartite graph $\mathcal{G}$ is actually a *heterogeneous* graph containing multiple types of nodes (user, item and bundle nodes) and multiple types of edges (user-item, user-bundle, bundle-item edges). Inspired by the *Relational Graph Convolutional Network* (R-GCN) model [15], we take the heterogeneous properties into account for our problem, and extends the GCN model to relational graphs, which could be considered as directed and labeled heterogeneous graphs. In our user-item-bundle tripartite graph setting, we consider three kinds of relations, i.e., the user-item interaction relation, the user-bundle interaction relation, and the bundle-item interaction relation, which consists of six relational edge[3]. The propagation rule for calculating the forward-pass update of an node i in a relational graph is as follows:

$$\mathbf{h}_i^{l+1} = \sigma\left(\mathbf{W}^l\mathbf{h}_i^l + \sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_i^r} \frac{1}{c_{i,r}} \mathbf{W}_r^l \mathbf{h}_j^l\right) \qquad (7)$$

where $\mathcal{N}_i^r$ denotes the set of neighbor indices of node i under relation $r \in \mathcal{R}$, $\mathbf{W}^l \in \mathbb{R}^{d_0 \times d}$ denotes a trainable weight matrix. $c_{i,r}$ is a problem-specific normalization constant that can either be learned or chosen in advance (we use $c_{i,r} = |\mathcal{N}_i^r|$ in this work).

---

[3]$\mathcal{R}$ contains relations both in canonical direction (e.g. user->item) and in inverse direction (e.g. item->user).

3.3 Link Prediction Layer

After the iterative diffusion process propagated with L layers, we obtain multiple representations for user node u, namely $\{\mathbf{h}_u^1, ..., \mathbf{h}_u^L\}$. The hidden representations obtained in different layers emphasize the messages passed over different connections or search depth in the graph, which makes them have different contributions in reflecting user preference. As such, we concatenate them to constitute the final user representations. Likewise, we can obtain the bundle and item representations by concatenating the bundle and item node representations $\{\mathbf{h}_b^1, ..., \mathbf{h}_b^L\}$, $\{\mathbf{h}_i^1, ..., \mathbf{h}_i^L\}$ learned by different layers.

$$\mathbf{h}_u = \mathbf{h}_u^1 || \cdots || \mathbf{h}_u^L, \quad \mathbf{h}_i = \mathbf{h}_i^1 || \cdots || \mathbf{h}_i^L, \quad \mathbf{h}_b = \mathbf{h}_b^1 || \cdots || \mathbf{h}_b^L \quad (8)$$

where $||$ is the concatenation operation. In our experiments, we set $L = 2$ since we found that stacking more than two convolutional layers did not improve performance.

The introduction of item information as a bridge role can make the model have a richer representation ability, which can be verified from the following experiments. Here, we simultaneously model user preferences for items and bundles, expecting their prediction performance to be mutually reinforced. Thus, with the final representations of users, items and bundles, we concatenate the latent vector representations of user u and item i as $\mathbf{h}_u || \mathbf{h}_i$, and feed them into a two-layer fully connected network (multilayer perceptron, MLP) to predict the preference $\hat{p}_{ui}$ of user u to item i, and feed $\mathbf{h}_u || \mathbf{h}_b$ into another two-layer MLP to predict the preference $\hat{p}_{ub}$ of user u to bundle b.

$$\hat{p}_{ui} = \text{sigmoid}\left(\mathbf{W}_i^2 \text{ReLU}\left(\mathbf{W}_i^1[\mathbf{h}_u || \mathbf{h}_i] + \mathbf{b}_i^1\right) + \mathbf{b}_i^2\right) \qquad (9)$$

$$\hat{p}_{ub} = \text{sigmoid}\left(\mathbf{W}_b^2 \text{ReLU}\left(\mathbf{W}_b^1[\mathbf{h}_u || \mathbf{h}_b] + \mathbf{b}_b^1\right) + \mathbf{b}_b^2\right) \qquad (10)$$

where $\mathbf{W}_i^1, \mathbf{W}_b^1 \in \mathbb{R}^{d_1 \times 2d_0}$ and $\mathbf{W}_i^2, \mathbf{W}_b^2 \in \mathbb{R}^{d_2 \times d_1}$ are corresponding weight matrices, $\mathbf{b}_i^1, \mathbf{b}_b^1 \in \mathbb{R}^{d_1}$ and $\mathbf{b}_i^2, \mathbf{b}_b^2 \in \mathbb{R}^{d_2}$ are corresponding bias, respectively.

3.4 Model Training

3.4.1 Loss Function. To train our BundleNet model, we adopt the Bayesian Personalized Ranking (BPR) [13] loss function. As a pairwise learning framework, BPR is an very pervasive personalized

ranking criterion used in recommender systems and information retrieval community. It is based on the triplets data $\{u, p, n\}$, and the semantics is that user u is assumed to prefer positive item p over negative item n:

$$L_{BPR}(u, p, n) = -\ln \sigma(\hat{p}_{up} - \hat{p}_{un}) + \lambda ||\Theta||_2^2 \quad (11)$$

where Θ denotes model parameters. $L2$ regularization is applied to prevent overfitting and λ controls the regularization strength.

3.4.2 Multi-Task Learning. By enforcing a common intermediate representation, Multi-Task Learning (MTL) can lead to better generalization and benefit all of the tasks, if the different problems are sufficiently related. This is obviously applicable in our scenario when we consider the user's preferences for items and bundles at the same time. In our multi-task learning framework, we construct two kinds of triplets , i.e., user-item triplets $\{u, i^+, i^-\}$ and user-bundle triplets $\{u, b^+, b^-\}$, corresponding to two loss functions:

$$L_1 = L_{BPR}(u, i^+, i^-), \quad L_2 = L_{BPR}(u, b^+, b^-) \quad (12)$$

For triplets $\{u, i^+, i^-\}$, we first sample a user u, then sample the positive item i^+ from the bundles which u have interaction history with, and a paired negative item i^- from the rest of items. The similar process is performed for triplets $\{u, b^+, b^-\}$. In the experiments, we first use user-item interaction to minimize L_1 for pre-training, and then continue training with the bundle information to minimize $L2$ until convergence. An alternative strategy is to execute two gradient steps in turn to minimize L_1 and $L2$ [3].

3.4.3 Label Leakage Issue. We notice that the usual GCN-like model has a label leakage issue when it is used to solve the link prediction problem, which is also noted by [18]. Specifically, according to the design principle of the GCN, each node aggregate all neighbor information to update its self-representation. As shown in Figure 5(a), for example, when we want to predict an edge $e = (u_1, b_2)$, we have to learn the representation of both node u_1 and b_2. However, as the neighbor of u_1, we will aggregate the information of b_2 (along with b_1 and b_3) when we update the representation of node u_1. Similarly, when we update the representation of node b_2, we will also use the information of u_1. This means that the model actually tries to learn such a mapping $f_\theta(e, \cdots) = e$, leading to the label leakage issue, although it is a bit implicit in the GCN framework. The reason for this issue is that, when applied to link prediction problem, the usual GCN training method involves all elements of the entire graph to participate in training simultaneously (predict all existing edges in the graph, including e of course).

To avoid the label leakage issue, we need to make sure that the edge information (e.g., $e = (u_1, b_2)$) is not used when predicting the edge itself. Although the dropout technology can alleviate this, however, it does not essentially address the problem. Inspired by the training strategy in [18], we adapt the usual (vanilla) **full-batch** training method of GCN to the **mini-batch** setting in the context of link prediction, following a *sampling-deleting-predict* strategy. Instead of using all edges, at each training iteration step, we proceed as follows: first, we sample a batch of edges from the training graph (denoted as the red lines in the Figure 5(b)), then we delete these sampled edges from the graph to ensure that they will not participate in the neighborhood aggregation operation during the

training process. Finally, we perform the usual GCN training procedure on the modified graph, but only to predict those sampled (and deleted) edges, instead of all of the edges in the graph. With the mini-batch training method, we observe a substantial boost in link prediction performance, which can be observed in the compared results in following experiments.
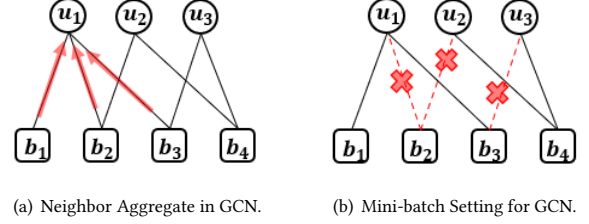

(a) Neighbor Aggregate in GCN.   (b) Mini-batch Setting for GCN.

Figure 5: The Label Leakage Issue for Link Prediction.

3.5 Model Inference

After the training is completed, in the inference stage, the test user id u, the bundle id b and all item ids $\{i | i \in b\}$ within the bundle are fed into the model. Then, we can obtain the user's preference prediction for the bundle $\hat{p}_{ub}$ and for the included items $\{\hat{p}_{ui} | i \in b\}$. The final prediction of user's preference for the bundle is:

$$\hat{p} = \hat{p}_{ub} + \frac{1}{|b|} \sum_{i \in b} \hat{p}_{ui} \quad (13)$$

For a newly released bundle, we could set $\hat{p}_{ub}$ to 0, and get the final preference prediction of the bundle just based on the user's prediction of the item, alleviating the cold start problem.

4 EXPERIMENTS

4.1 Datasets

we evaluate all the models on three public datasets and one industrial dataset. The *Steam* dataset is collected from the Steam[4] video game distribution platform by [11], where each bundle consists of several video games. The *NetEase* dataset, provided by the work in [2], is crawled from the Netease Cloud Music[5], which enables users to construct the bundle (a list of songs) with a specific theme. The *Youshu* dataset introduced by [3] is constructed by crawling data from a book review site Youshu[6], where each bundle is a list of books constructed by website users. Finally, the *Justice* dataset is collected from the mobile game *Love is Justice* developed by Netease Games, where bundles are made up of props (virtual items) in the game. The statistics of datasets are briefly shown in Table 1.

4.2 Baselines

- **BPR** [13]: This model is the basic pairwise ranking algorithm based on implicit feedback. We learn a BPR baseline model by user-bundle interactions, and optimize the BPR ranking loss under the matrix factorization framework.

---

[4] https://store.steampowered.com/
[5] https://music.163.com/
[6] https://www.yousuu.com/

Table 1: Summary Statistics of the Datasets

Datasets	# users	# bundles	# items	# user-bundle (density)	# user-item (density)	# bundle-item (density)
Steam	29,634	615	2,819	87,565 (0.48%)	902,967 (1.08%)	3,541 (0.20%)
Youshu	8,039	4,771	32,770	51,377 (0.13%)	138,515 (0.05%)	176,667 (0.11%)
NetEase	18,528	22,864	123,628	302,303 (0.07%)	1,128,065 (0.05%)	1,778,838 (0.06%)
Justice	25,470	234	278	117,873 (1.98%)	379,384 (5.36%)	483 (0.74%)

- **BundleBPR** [16]: This is a bundle BPR model which makes use of the parameters learned through the item BPR.
- **DAM** [3]: This model is specially designed for the bundle recommendation, which jointly models user-bundle interactions and user-item interactions in a multi-task manner.
- **GCN-Bi** [9]: The GCN model learned on the user-bundle bipartite graph with usual full-batch training method.
- **GCN-Bi-B** [9]: The GCN-Bi model with our mini-batch training method introduced in section 3.4.3.
- **GCN-Tri**: The adapted GCN learned on our user-item-bundle tripartite graph with usual full-batch training method.
- **GCN-Tri-B**: The GCN-Tri mdoel with our mini-batch training method introduced in section 3.4.3.
- **BundleNet**: The adapted Relational GCN model learned on the user-item-bundle tripartite graph.
- **BundleNet-B**: The BundleNet model with our mini-batch training method introduced in section 3.4.3.

4.3 Metrics

We adopt the *leave-one-out* evaluation in our experiments, which has been widely used in the literature [6, 13]. For each user, we randomly hold-out one of her bundle interactions for testing and the remaining data for training. Since it is too time-consuming to rank all bundles for all users during the evaluation, we followed the common strategy [6] that randomly samples 99 bundles that are not interacted by the user as negative samples, ranking the test bundle among the 99 bundles. To evaluate the recommendation performance of bundle models, we use several widely adopted ranking metrics in information retrieval: Recall, Mean Reciprocal Rank (MRR) and Normalized Discounted Cumulative Gain (NDCG). Specifically, the Recall@K measures the number of positive bundles presenting within the top-K recommendation list. The MRR@K considers the rank position of the first positive bundle for the user within the top-K recommendation list. And the NDCG@K accounts for the position of the positive bundles by additionally considering the discount factor at lower ranks.

4.4 Implementation

All models were implemented in PyTorch[7] with the Adam optimizer [8]. We also employ early stopping and dropout techniques to prevent over-fitting. The embedding size d is fixed to 32 for all models. The hidden sizes d_0, d_1, d_2 are set to 64, 256, 128 respectively. The batch size for edge sampling is fixed to 1024. We apply grid search for tuning the hyper-parameters of the models: the learning rate is tuned amongst {0.0001, 0.0005, 0.001, 0.005, 0.01}, the coefficient

---

[7]https://pytorch.org/

of $L2$ regularization is searched in $\{10^{-5}, 10^{-4}, ..., 1, 10^1\}$, and the dropout ratio in {0.0, 0.1, ..., 0.5}. The set of possible hyper-parameter values was determined on early validation tests using subsets of the datasets that we then discarded from our analyses.

4.5 Results and Analysis

We conduct extensive experiments on the datasets with the above benchmark methods to evaluate our model. We use 80% of the data as training set to learn model parameters, 10% as validation data to tune hyper-parameters and the rest 10% as test set for final performance comparison. We repeat this procedure 10 times and report the average ranking values, which is summarized and shown in Table 2. We can find that our proposed method outperforms the baseline methods significantly in all datasets. From the experimental result, we also have several interesting findings listed as follows:

- The models of utilizing user-item interactions always outperform the models of not using this information, e.g., *BundleBPR* is better than traditional *BPR* and *GCN-Tri* is better than *GCN-Bi*. This result is obviously in line with our expectations and verifies the effectiveness of introducing item interaction information in the bundle recommendation problem. This shows that leveraging the items as bridge signal/nodes to learn the representations of the users and/or bundles can alleviate the data sparsity problem.
- When considering modeling the bundle recommendation as a link prediction problem on the graph, models with mini-batch training method introduced in section 3.4.3 always outperform the models without using this information, e.g., the *GCN-Bi-B* and *BundleNet-B* is better than *GCN-Bi* and *BundleNet*, respectively. We think the phenomenon is caused by the label leakage issue introduced above, and can be effectively alleviated through the mini-batch training trick. We believe that such comparison results bring us some useful inspirations, when using the GCN-like model for link prediction tasks.
- Our proposed model *BundleNet* performs better than the state-of-the-art bundle recommendation method *DAM*, which proves the effectiveness of modeling bundle recommendation as the link prediction problem in the user-item-bundle tripartite graph. Moreover, the *BundleNet*/*BundleNet-B* is slightly superior than the *GCN-Tri*/*GCN-Tri-B* for most datasets, which indicates that the heterogeneous characteristics of the user, item and bundle nodes and their interactions usually should not be ignored. However, in the *NetEase* dataset, it is a bit worse. We guess that this is related to the characteristics of the data set, and it is worth further exploration.

Table 2: Comparison of Results (for GCN-related models, including our model BundleNet, model names with and without the -*B* suffix indicate that the mini-batch training method and the normal full-batch training method is used, respectively).

	Steam			Youshu			NetEase			Justice		
	Recall@5	MRR@5	NDCG@5	Recall@5	MRR@5	NDCG@5	Recall@5	MRR@5	NDCG@5	Recall@5	MRR@5	NDCG@5
BPR	0.9712	0.8002	0.8437	0.5409	0.3781	0.4278	0.3532	0.2086	0.2198	0.6735	0.4707	0.5223
BundleBPR	**0.9818**	**0.8219**	**0.8594**	0.5912	0.3923	0.4408	0.4677	0.2765	0.3342	0.6925	0.5022	0.5482
DAM	0.9792	0.8016	0.8467	**0.5996**	**0.4049**	0.4532	0.4109	0.2424	0.2841	0.7117	0.4764	0.5349
GCN-Bi	0.9793	0.8069	0.8508	0.5753	0.3776	0.4267	0.3493	0.2037	0.2397	0.5578	0.3563	0.4061
GCN-Bi-B	0.9794	0.8106	0.8535	0.6001	0.4006	0.4503	0.4275	0.2597	0.3013	0.7427	0.4985	0.5594
GCN-Tri	0.9797	0.8012	0.8465	0.5893	0.3915	0.4408	0.3641	0.2138	0.2509	0.5718	0.3651	0.4172
GCN-Tri-B	0.9788	0.8092	0.8524	0.5924	0.3959	**0.4548**	**0.5252**	**0.3231**	**0.3732**	**0.7618**	**0.5193**	**0.5797**
BundleNet	0.9788	0.8108	0.8536	0.5927	0.3962	0.4452	0.3579	0.2119	0.2481	0.5754	0.3742	0.4162
BundleNet-B	**0.9848**	**0.8859**	**0.9112**	**0.6241**	**0.4247**	**0.4668**	**0.5142**	**0.3114**	**0.3616**	**0.7705**	**0.5545**	**0.5807**

4.6 Ablation Study

In addition to the user-item-bundle tripartite graph, there are several designs involved in our model: the Relational GCN (**REL**) to account for heterogeneous properties of graph nodes, the multi-task learning (**MTL**) framework to model user's preferences for items and bundles simultaneously, and the mini-batch training (**MBT**) method to solve the label leakage issue. To evaluate the effectiveness of these major designs, we carried out ablation studies as shown in Figure 6. The result demonstrates that these designs show different improvements for different datasets. For example, the **MBT** is crucial for NetEase and Justice, while both **REL** and **MBT** is beneficial to Steam. Meanwhile, Youshu is not very sensitive to these designs, which means its performance improvement mainly depends on the basic tripartite graph design.
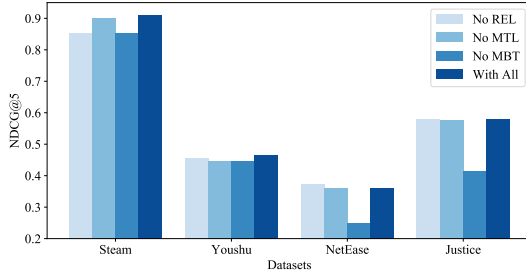

Figure 6: Performance Comparison of Major Designs.

4.7 Online Deployment

The proposed recommendation model has been deployed in production for more than one year in *Love is Justice* developed by Netedse Games. In this section, we briefly give some implementation details of the bundle recommendation pipeline in the online game, as shown in Figure 7.

- **Data Module.** The data module is responsible for data storage and preprocessing tasks. The historical interaction data between users and items as well as bundles within a period of time is used to generate training data.
- **Training Module.** We train and update our recommendation model on a daily basis. Since retraining the model from
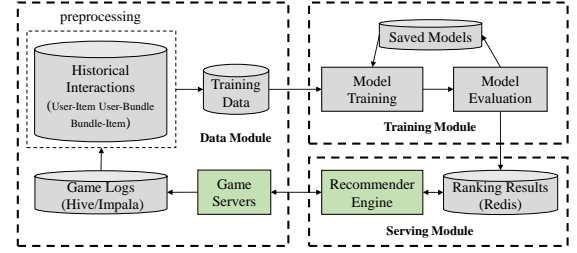

Figure 7: The Overview of Bundle Recommendation Pipline.

scratch every time is computationally time-consuming, a better solution is to use the previously saved model as pre-training, and fine-tune the model on new data every day, which leads to faster convergence of model training.

- **Serving Module.** Once the model is trained and verified, we predict the preference scores, which are obtained by running a forward inference pass over the model, of all the bundles for all users. Then, the personalized bundles are ranked from the highest scores to the lowest, and the result is stored into the database for quick retrieval.

Figure 8(a) gives the online performance within a period of nearly three months of the presented approach compared with a heuristic method in production, which is a combination of handcrafted recommendation rules. We can find that our proposed method always outperforms the heuristic method in online A/B testing. According to our analysis of purchase statistics, the launch of the model yields more than 60% improvement on conversion rate (CVR) of bundles on average, and a relative improvement of more than 15% in terms of gross merchandise volume (GMV).

For an in-depth analysis of the improvement, we calculate the conversion rate of most representative bundles with different prices separately. As shown in Figure 8(b) (the specific price values are properly processed on the scale for privacy considerations), we can find that the main reason for the improvement lies in the accurate recommendation of high-priced bundles. These bundles often contain more valuable items that are very attractive to players interested in them. Different from the lower-priced bundles which usually only contain common items, the high-priced bundles are highly personalized which leaves room for improvement. We also
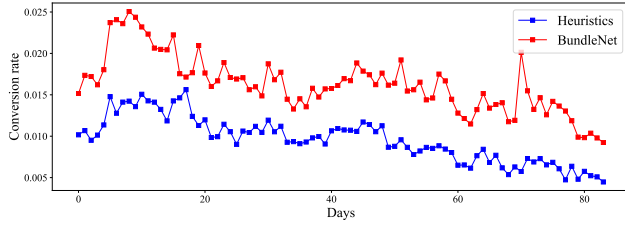
(a) Comparison of Conversion Rate (from left to right arranged by date).
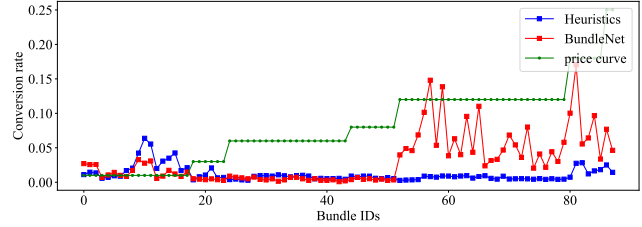
(b) Comparison of Conversion Rate of Bundles with Different Prices (from left to right arranged in ascending bundle prices).

Figure 8: Online Performance of BundleNet on the Game *Love is Justice*.

noticed that the purchase rate of low-priced bundles is higher than that of middle-priced bundles. We speculate that the types of items included in these bundles are not much different, but low-priced bundles are more appealing in price.

5 RELATED WORK

In the field of recommendation, there have been several efforts to solve the problem of bundle recommendation. The *List Recommendation Model* (LIRE) solves the recommendation problem of user-generated item lists based on a latent factor-based BPR model, which takes into consideration users' previous interactions with both item lists and individual items. *Embedding Factorization Model* (EFM) [2] is proposed to jointly model the user-item and user-list interactions, which combines two types of latent factor models: BPR [13] and word2vec [10]. Also building upon the BPR model, [11] tries to recommend existing bundles to users on the basis of their constituent items, as well as the more difficult task of generating new bundles that are personalized to a user via the bundle-level BPR model, which makes use of the parameters learned through the item-level BPR model. *Deep Attentive Multi-Task* DAM [3] model designs a factorized attention network to aggregate the embeddings of items within a bundle to obtain the bundle's representation, while jointly model user-bundle interactions and user-item interactions in a multi-task manner to alleviate the scarcity of user-bundle interactions. Some other related efforts include [1, 5, 7, 12, 14].

6 CONCLUSION AND FUTURE WORK

In this paper, we target at a practical but less explored recommendation problem named bundle recommendation. Different from the traditional item recommendation problem, it aims to recommend a *bundle* (i.e., a combination of items) rather than the individual item to the target user. To tackle this specific recommendation problem instance in the context of the virtual mall in online games, we highlight the challenges and formalize it as a link prediction problem on a user-item-bundle tripartite graph, which is constructed from the historical interactions, and solve it within an end-to-end graph neural network framework. Extensive offline and online experiments demonstrate the effectiveness of the presented method.

REFERENCES

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

[2] Da Cao, Liqiang Nie, Xiangnan He, Xiaochi Wei, Shunzhi Zhu, and Tat-Seng Chua. 2017. Embedding factorization models for jointly recommending items and user generated lists. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 585–594.

[3] Liang Chen, Yang Liu, Xiangnan He, Lianli Gao, and Zibin Zheng. 2019. Matching user with item set: collaborative bundle recommendation with deep attention network. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2095–2101.

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

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

[6] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[7] Yun He, Jianling Wang, Wei Niu, and James Caverlee. 2019. A Hierarchical Self-Attentive Model for Recommending User-Generated Item Lists. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1481–1490.

[8] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[9] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[11] Apurva Pathak, Kshitiz Gupta, and Julian McAuley. 2017. Generating and personalizing bundle recommendations on steam. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1073–1076.

[12] Shuyao Qi, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. 2016. Recommending packages to groups. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 449–458.

[13] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. 452–461.

[14] Oren Sar Shalom, Noam Koenigstein, Ulrich Paquet, and Hastagiri P Vanchinathan. 2016. Beyond collaborative filtering: The list recommendation problem. In *Proceedings of the 25th international conference on world wide web*. 63–72.

[15] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*. Springer, 593–607.

[16] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[17] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A neural influence diffusion model for social recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*. 235–244.

[18] Jiani Zhang, Xingjian Shi, Shenglin Zhao, and Irwin King. 2019. STAR-GCN: stacked and reconstructed graph convolutional networks for recommender systems. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 4264–4270.