Generating and Personalizing Bundle Recommendations on Steam

Apurva Pathak  
University of California, San Diego  
apathak@ucsd.edu

Kshitiz Gupta  
University of California, San Diego  
ksg005@ucsd.edu

Julian McAuley  
University of California, San Diego  
jmcauley@cs.ucsd.edu

ABSTRACT

Many websites offer promotions in terms of bundled items that can be purchased together, usually at a discounted rate. ‘Bundling’ may be a means of increasing sales revenue, but may also be a means for content creators to expose users to new items that they may not have considered in isolation. In this paper, we seek to understand the semantics of what constitutes a ‘good’ bundle, in order to recommend existing bundles to users on the basis of their constituent products, as well as the more difficult task of generating new bundles that are personalized to a user. To do so we collect a new dataset from the Steam video game distribution platform, which is unique in that it contains both ‘traditional’ recommendation data (rating and purchase histories between users and items), as well as bundle purchase information. We assess issues such as bundle size and item compatibility, and show that these features, when combined with traditional matrix factorization techniques, can lead to highly effective bundle recommendation and generation.

1 INTRODUCTION

The basic goal of a Recommender System is to understand relationships between users and the items they consume. Typically, this is cast as estimating compatible user/item (or item/item) pairs, and surfacing these as recommendations. An inherently more challenging task is to understand the semantics that describe relationships between sets of items, in terms of what factors make them mutually compatible and mutually desirable to a user. One area where such semantics are important is for recommending bundles, or sets of items that can be simultaneously co-purchased.

In this paper, we seek to apply recommender systems techniques to bundle generation and recommendation tasks. Bundles are ubiquitous on e-commerce platforms, though the semantics of how to compose and recommend them have rarely been studied. One reason for the relative scarcity of work in this area may simply be the lack of suitable data to explore users’ interactions with bundles. Here we contribute a dataset extracted from the Steam video game distribution platform, which offers detailed information both in terms of user/item interactions (including what games users purchased, whether they played them or not after purchase, and how they rated them), as well bundle promotions, and sufficient information to extract what bundles were purchased by each user.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGIR ’17, August 07-11, 2017, Shinjuku, Tokyo, Japan
© 2017 ACM. 978-1-4503-5022-8/17/08...$15.00
DOI: http://dx.doi.org/10.1145/3077136.3080724

1 All code and data is available at http://cseweb.ucsd.edu/~jmcauley/

Together, the features of our data allow us to assess how the semantics of bundle recommendation differ from traditional item recommendation problems. We build on traditional techniques for item-to-user recommendation, in order to assess the extent to which items that are bundled together should be mutually compatible, desirable, or diverse in their features. Beyond personalized bundle recommendation, we can use the learned objective to generate new bundles, in order to surface personalized promotions to a user.

2 RELATED WORK

We build upon latent factor models, and in particular Bayesian Personalized Ranking (BPR) [11], which is trained using implicit feedback (i.e., purchases vs. non-purchases) in order to estimate rankings of items that are likely to be interacted with.

Also related are systems that recommend items to groups of users (i.e., recommending items to sets of users, as opposed to recommending sets of items to users). For example, group recommendation can be addressed by developing techniques that aggregate preferences of individuals within groups. [2, 3, 5, 7]. Bundling products for groups is also considered in [10], though their main focus is item-to-group compatibility. Although the formulations vary, such methods essentially work by maximizing item relevance while minimizing disagreements between group members.

We also build on systems that consider item aggregate diversity. Adomavicius and Kwon [1] study the importance of balancing accuracy and aggregate diversity for ranking tasks, and Nieman and Wolpers [9] study the relationship between aggregate diversity and item co-occurrence.

Finally, given that our goal is to generate sets of items, our work is related to papers that study basket recommendation and item compatibility. This includes both classical works on market baskets (and itemset/association rule mining), as well as more modern works that learn substitute and complement relationships [6, 13]. Our work differs from typical bundling scenarios (e.g. grocery shopping) where similar sets of items can be recommended repeatedly. Few recent works consider related forms of ‘bundling’ [4, 12, 14], though differ in terms of problem formulation, or lack actual bundle sales data and therefore rely on heuristics for evaluation.

3 DATASET AND ANALYSIS

We use data from the Steam video game distribution network for training purposes.1 We focus on the Australian Steam community, by crawling users from the ‘GameAUs’ community. In total, this resulted in a network of 88,310 gamers and 10,978 games they purchased. Basic statistics are shown in Table 1.

Among the 615 bundles available on Steam at the time of our experiments, we found that 29,634 gamers (around 33%) purchased
We use a graph sampling technique to create a balanced training set. We first provide methods for personalized ranking (as described below), then learn users’ preferences over individual items. To qualitatively poor results. To overcome this problem of data skew, we use a graph sampling algorithm [8] that creates D_item and D_bundle such that their negative items and bundles follow the same degree distribution as their positive items and bundles. What this means in practice is that it is not possible to distinguish positive versus negative items/bundles based only on their popularity, forcing the model to learn a richer notion of compatibility.

### 5.1 Bayesian Personalized Ranking (BPR)

The goal of BPR is to derive a personalized ranking >_u over items (or bundles). To model the ranking, we assume an estimator x : U × I → R encoding the compatibility between a user and an item, which is used to define the ranking

\[ i_p > i_n \iff \hat{x}_{u,i_p} > \hat{x}_{u,i_n}. \]

The optimization criterion for BPR, BPROpt, as derived in [11] is:

\[ \text{BPROpt}(\theta) = \sum_{(u,i_p,i_n) \in D} \log(\sigma(\hat{x}_{u,i_p}(\theta) - \hat{x}_{u,i_n}(\theta))) - \lambda \|\theta\|^2 \]

where, \( \sigma \) is the sigmoid function, \( \theta \) is the parameter vector of the compatibility function, \( D \) represents the training set, \( \lambda \) is the regularization hyper-parameter, \( \hat{x}_{u,i_p} \) and \( \hat{x}_{u,i_n} \) represent compatibility estimates that the user \( u \) would purchase item \( p \) and \( n \) respectively. Recall that \( i_p \) and \( i_n \) are a positive and a negative item, so the expression \( \sigma(\hat{x}_{u,i_p}(\theta) - \hat{x}_{u,i_n}(\theta)) \) essentially captures the probability that the purchased item is correctly identified as being more compatible than the non-purchased one.

We use different predictors \( \hat{x}_{u,i} \) and \( \hat{x}_{u,b} \) when considering item and bundle recommendation, as described below.

#### 5.1.1 Item BPR

The estimator function for the item BPR model is based on matrix factorization:

\[ \hat{x}_{u,i} = \beta_i + P_u \cdot Q_i \]

where \( \beta_i \) is an item parameter, and \( P_u \) and \( Q_i \) are \( k \)-dimensional latent parameter vectors for user \( u \) and item \( i \) respectively, to be learned by optimizing BPROpt over \( D_{item} \).

#### 5.1.2 Bundle BPR

The bundle BPR model makes use of the parameters learned through the item BPR model to estimate the preference of a user toward a bundle as shown below:

\[ \hat{x}_{u,b} = \frac{1}{|B_k|} \sum_{i \in B_k} [\kappa \beta_i + (\mu P_u) \cdot (\omega Q_i)] + C_{ck} + N_p \]

where \( \beta \), \( P \), and \( Q \) are learned from the item BPR model. \( \mu \) and \( \omega \) are \( k \times k \) dimensional matrix adjustment parameters for \( P \) and \( Q \) respectively. \( c_{ck} \) represents the bundle correlation, which is the mean pair-wise Pearson correlation of the items, represented using their latent features \( Q_i \), present in the bundle. \( N \) is a \( \max_{B_k} |B_k| \) dimensional parameter that rewards or penalizes bundles of certain sizes (to prevent the system from generating arbitrarily large bundles). The remaining parameters are scalars that trade-off various terms. All parameters are learned by optimizing BPROpt on \( D_{bundle} \).

---

### Table 1: Australian Community Data

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users</td>
<td>88,310</td>
</tr>
<tr>
<td>Total games</td>
<td>10,978</td>
</tr>
<tr>
<td>Total game purchases</td>
<td>902,967</td>
</tr>
<tr>
<td>Total bundles</td>
<td>615</td>
</tr>
<tr>
<td>Total bundle purchases</td>
<td>87,565</td>
</tr>
<tr>
<td>Users who purchased at least one bundle</td>
<td>29,634</td>
</tr>
<tr>
<td>Games that appear in at least one bundle</td>
<td>2,819</td>
</tr>
<tr>
<td>Average bundle size</td>
<td>5.73</td>
</tr>
</tbody>
</table>

---

2The scheme itself is a simple randomized ‘rewiring’ procedure on the original user-item or user-bundle purchase graphs, which has previously been used to correct skew when training recommender systems [6].
5.1.3 Cold Bundle Problem. We define ‘cold’ bundles as those which contain at least one item \( i \in I \) which is not observed in any of the existing bundles in the dataset. Because of the way we have trained the bundle BPR model (as a function of parameters of the item BPR model), our model is robust to cold bundles, i.e. it can rank any bundle containing items \( I' \subset I \).

5.2 Evaluation

We compute the AUC to evaluate both item BPR and bundle BPR. The AUC is given by:

\[
AUC = \frac{1}{|T|} \sum_{(u,p,n) \in T} \delta(\hat{x}_{u,i_p} - \hat{x}_{u,i_n} > 0)
\]

where \( \delta \) is the indicator function and \( T \) is the fraction of the data withheld for testing. In other words, we are counting the fraction of times the model correctly ranks \( p \) higher than \( n \).

6 PERSONALIZED BUNDLES GENERATION

So far, we have considered recommending bundles that already exist within the system. Next, we present a greedy algorithm that uses the learned parameters to generate new bundles.

6.1 Greedy Algorithm

The algorithm is described below. We start with an initial bundle (of size \( S = 3 \)) and select \( k = 10 \) neighbors in every iteration. The size of the neighbor set, \( k \), is inversely related to the aggregate diversity of the generated bundle, i.e., as \( k \) increases the method will tend to favor popular items in generated bundles.

(1) Start with a bundle \( b \) containing \( S \) randomly-chosen items.

(2) Randomly select a set of \( k \) items \( I_n \subset I \setminus \{i \in b\} \) and form a set of new bundles \( B_b \) by adding, deleting, and substituting items from \( I_n \).

(3) Let \( b_k \) be the most preferred bundle (by a user \( u \)) among all bundles \( B_b \) and \( P \) be the corresponding preference score, i.e., \( b_k = \arg \max_{b \in B_b} (x_{u,b} - \hat{x}_{u,b}) \) and \( P = \max_{b \in B_b} (x_{u,b} - \hat{x}_{u,b}) \).

(4) If \( P > 0 \), then accept \( b_k \) as the new bundle, otherwise accept \( b_k \) with diminishing probability (following an annealing schedule).

(5) Repeat steps 2 to 4 until convergence.

6.2 Evaluation

We evaluate the generated bundles for a set of users using two criteria. First, we consider how the generated bundle ranks compared to existing bundles, according to our bundle BPR model. Second, we consider the aggregate diversity of the generated bundles, in order to assess the coverage of items within the system:

\[
\text{aggregate diversity} = \frac{\# \text{of distinct items across generated bundles}}{\# \text{of items}}
\]

The former of these evaluation measures is a simple sanity check to ensure the greedy approach finds local minima with high compatibility; the latter is a qualitative assessment to assess whether a diverse variety of bundles are recommended.

7 EXPERIMENTS

We consider items that occur in at least one existing bundle, resulting in an item set of size 2,819. Using the sampling method described in Section 5, we create bundle and item data samples of sizes 1,016, 646 and 26, 717, 059 (respectively) which we partition into 70%/10%/20% training/validation/test splits. We report performance on the test set for the model that performs best on the validation set.

7.1 Bundle Ranking

We compare the performance of our bundle ranking model against several baselines: (1) Regular BPR (BR); (2) Bundle BPR using item features (I-BPR); (3) Our ‘vanilla’ model without bundle size and bundle correlation (BR); and (4) BR with bundle size (BR + N). Finally, we report (5) our proposed model including size and bundle correlation (BR + N + C).

Performance in terms of the AUC is shown in Table 2 (left). BPR performs well but cannot be used for bundle generation as it uses only bundle features and not item features. I-BPR overcomes this limitation by using the mean over the item features to estimate the preference of a user towards a bundle at the cost of reduced AUC. Our vanilla model (BR)—without bundle correlation and bundle size terms—has AUC close to that of BPR while using only item features. The main difference between BR and BPR is that the parameters \( \beta, \beta' \) and \( \gamma \) in BR are constants learned from the item BPR model. When we add bundle correlation and bundle size terms in BR it substantially outperforms BPR.

Cold Bundles. In Section 5.1.3, we suggested that our model is robust to the cold bundle problem. To validate this we create a reduced item set, \( I_r \), by randomly removing 219 items from \( I \) and resampling the training data, \( D_{bundle} \), such that bundles only contain items present in the reduced item set, \( I_r \), where \( |I_r| = 2600 \). The test set then consists of triplets \((u, b_p, b_n)\) such that \( b_p \) is a ‘cold’ bundle (containing at least one of the removed items) and \( b_n \) is not. It should be noted that the data for the item BPR model remains unaltered as we are dealing with the cold bundle problem and not the cold item problem. We observe in Table 2 (right) that when we test our bundle BPR model on the test set we observe only a modest reduction in AUC in spite of the presence of cold bundles.

7.2 Bundle Generation

We generate new bundles using the greedy algorithm described in Section 6.1 for 1000 randomly selected users (Table 3). The BR model doesn’t have bundle size and bundle correlation as features and hence reduces to an average of item BPR features; for this model we set the minimum bundle size to 2, otherwise the model tends to

<table>
<thead>
<tr>
<th>Model</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPR</td>
<td>0.8624</td>
</tr>
<tr>
<td>I-BPR</td>
<td>0.82212</td>
</tr>
<tr>
<td>BR</td>
<td>0.8519</td>
</tr>
<tr>
<td>BR + N</td>
<td>0.89447</td>
</tr>
<tr>
<td>BR + N + C</td>
<td>0.90276</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR</td>
<td>0.81203</td>
</tr>
<tr>
<td>BR + N</td>
<td>0.81928</td>
</tr>
<tr>
<td>BR + N + C</td>
<td>0.84669</td>
</tr>
</tbody>
</table>
generate bundles of size 1 (as the maximum compatibility is always greater than or equal to the average). Once we add the bundle size as a feature we start to observe a range of bundle sizes (generally between 3 and 8), though this has the effect of increasing the average rank. Average rank and aggregate diversity both improve slightly after including bundle correlation as a feature.

Our final model with bundle correlation is able to generate bundles that are qualitatively similar to those preferred by Steam users. Figure 1 shows a t-SNE embedding of latent representations for all items such that similar items are close to each other, as well as a sample of real and generated bundles. Our generated bundles seem to consist primarily of three types of bundles: those in which all items are similar (e.g. the series of Half Life games); those consisting of multiple series collected together (e.g. multiple Bioshock games along with multiple Sid Meier games); and bundles of different games with similar types (e.g. Shadow Warrior, Grand Theft Auto, Assassin’s Creed, and The Witcher).

In general, bundles with correlated games (in terms of their item BPR representations) receive higher scores than those with uncorrelated or negatively correlated items (Figure 2). Each point in Figure 2a represents the mean bundle score ($\hat{x}_{u,b}$) for 500 users and bundle correlation, whereas each point in Figure 2b represents the bundle score and correlation of the top bundle generated for a user. Both figures suggest a positive relationship between bundle correlation ($c_b$) and bundle score ($\hat{x}_{u,b}$).

Surprisingly, we find that few of the generated (or real) bundles are particularly diverse; rather they tend to consist of closely related games. Indeed, our best models identified positive bundle correlation terms, indicating that diversity is a feature that is penalized when considering users’ preferences toward real bundles.

8 CONCLUSION

We have shown a method to generate and evaluate personalized bundle recommendation on the Steam video game subscriber network. We developed a bundle BPR model which used the trained features of an item recommendation model in order to learn personalized rankings over bundles. We showed that our model is robust to cold bundles, and that new bundles can be generated effectively via a greedy algorithm.

As future work, we intend to adapt these approaches to generate recommendations for user populations (rather than individuals), and to investigate the role that pricing (and in particular, discount rate) has on users’ preferences toward bundles.

REFERENCES