Recommending Game Bundles using Hyperlink Prediction

Jen Weng
Stanford University
Palo Alto, CA
zzweng@stanford.edu

1 INTRODUCTION

Recommender system has become an increasingly important component in modern world, as eCommerce becomes ubiquitous in our lives. Recent research on recommender systems has been mainly concentrating on improving the relevance or profitability of individual recommended items. But in reality, users are usually exposed to a set of items and they may buy multiple items in one single order. The businesses, on the other hand, would put out a discount on a set of items (a bundle) so that they can decrease the shipping costs, attract buyers who are going to place bulge orders. Hence, bundle recommendation is of great interest to us.

For this project, we will review the literature in the area of recommendation algorithms (especially bundle recommendation) as well as hyperlink prediction. As we explained previously, we will be looking at the recommendation problem as hyperlink prediction problem in a hypergraph. Note that we can represent the purchase network as a hypergraph (Figure 1). Then the proposed hyperlinks would tell us which items can be grouped together as a bundle.

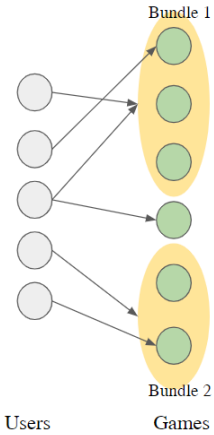

Figure 1: Example game purchase network modeled as a hypergraph. A hyperlink is a link between a user node and several game nodes grouped together (a bundle).

This report will first go over a bundle/item preference ranking algorithm introduced in (3). Although the AUC of this algorithm on the steam dataset is high, it only offers one way to look at the bundle desirability. Besides, generating a preference ranking for each user is an overkill in practice. In the remainder of the report, we will walk through our exploration to the question - How do we generate game bundles that are desirable to the users in general?

Preprint. Under review.

We will first use Coordinated Matrix Minimization (CMM), a recently proposed algo that uses an EM algorithm for predicting hyperlinks. The drawback of CMM is obvious. We are simply leveraging the distribution of the items/bundles and the purchases. Ideally, we want to be able to use the meta data for this dataset, as they can be informative. Hence, we will make use of the game metadata in our attempt of using GCN to propose good game bundles. We show that by formulating the hyperlink (bundle) prediction problem as a node classification problem on the dual of the original graph, we are able to encode the game features on every node and utilize graph neural networks to classify the nodes into binary categories (the two categories being whether the bundle is present in the dataset or not). We show that using GCN, we are able to train a model that has 90.1 percent accuracy in predicting the goodness of the bundle.

At the end of this report, we are able to answer the following questions:

1. Given two games (or bundles), which one does a certain user prefer?

2. Can we recommend some new bundles that may attract new purchases?

Being able to answer these questions is crucial to the eCommerce companies. If we can come up with a preference ranking among users, then we can adjust the advertising strategy for different users. If we are able to recommend some new bundles, then the existing users may buy those new bundles and we may also attract new users.

2 Related Work

The paper introduced for the Steam dataset (3) we will be using so it naturally becomes the first paper we review for this project. The authors use Bayesian Personalized Ranking (BPR) to rank the items/bundles. The goal of BPR is to derive a personalized ranking $>_u$ over items (or bundles). There are two estimators for item and bundle respectively. The parameters in the estimators are determined by an optimization problem such that the output of the estimator preserves the ranking of user preferences over items (or bundles). Their approach of dividing the recommendation problem into two sub-problems (ranking and bundle generation) is novel. Since they trained the bundle BPR model as a function of parameters of the item BPR model, their model is robust to cold bundles. Consequently, however, the evaluation could be largely influenced by the quality of the bundle generation algorithm. Moreover, the paper generates ranking for individual users instead of user populations, so it can be very expensive and might be an overkill in real-world applications. For instance, if the Steam platform simply wants to recommend some game bundles to a group of users, we certainly do not want to iterate over all users on the platform and generate individual rankings. Hyperlink prediction on the game networks would be much more efficient, since we know which users would want certain bundles once we have the predicted links.

In another paper (4), the authors proposed a new algorithm called Coordinated Matrix Minimization (CMM). At a high level, the algorithm starts with a matrix that contains the observed hyperlinks in the hypergraph, and alternately performs EM steps (nonnegative matrix factorization and least square matching in the vertex adjacency space of the hypernetwork) until a convergence threshold is satisfied or a maximum iteration number is reached. Upon convergence, they end up with a set of candidate hyperlinks with corresponding scores. Then, the candidate hyperlinks are ranked by their scores and the top ones are proposed as the new links. One experiment outlined in the paper resembles the task we are trying to do here. The dataset they used is a hyper network dataset with recipes. Each node is a material and each hyperlink is a combination of materials that constitute a dish. The goal is to predict new dishes based on the existing dishes. They rank all candidate hyperlinks with their scores and select the top 400 hyperlinks as predictions, and report how many of them are real missing recipes. This task is very similar to our setting where the nodes are individual items and the users, and each hyperlink is a combination of items that appeals to a user. This paper serves as a very good reference in terms of doing hyperlink prediction using a matrix factorisation-based technique. We will be using this method as the baseline for our project.

3 Dataset

We use the dataset from Professor Julian McAuley's lab website[1]. The dataset is free for download and it was crawled from an Australian Steam community 'GameAus'. There are 88,310 user profiles, 10,978 game profiles, 59,305 game reviews, and 615 game bundles in the dataset (3). In addition to the individual user's purchasing records, reviews from the Steam video game platform, and information about which games were bundled together, the dataset also includes the pricing information (prices for individual games in the bundle and the discount percentage). Not all the users have purchased game bundles. For the purpose of this project, we will filter out the users who have not purchased any bundles. After filtering, our dataset includes $29,634$ users (33% of all users), 615 bundles and there are 2819 games in those bundles. Of those 2819 games, the most purchased one got purchased by $14,000$ users.

For the main part of our project (Section 6), we will be using the bundle-item mapping dataset where we create a directed graph by inserting an edge from each bundle to each item. Note that we can think of each bundle as a hyperedge in a graph that is simply made of game nodes.

4 Exploratory Data Analysis

Bundle Purchases

In Figure 2, we present the distribution of the number of bundle purchases made by users and the distribution of the number of purchases for the bundles. We notice that although around 33% users have purchased bundles, the majority of them have purchased fewer than 10 bundles. Besides, majority of the bundles are only purchased by fewer than 50 users, which suggests that there are a lot of room for improvement in the bundle generation to increase the bundles popularity.

In addition, as pointed out in (3), the bundled items are more popular games: even though only 25% of items appear in any bundle, around 70% of purchases are over items that appear in a bundle. In practice, this means that users often purchase bundles containing some items they already own, suggesting a need to modify or personalize bundles toward a particular user.

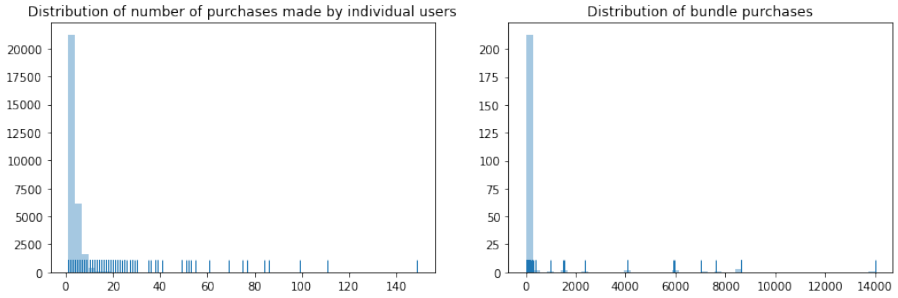

Figure 2: Left: Distribution of the number of bundle purchases made by all users. Right: Distribution of the number of purchases for all bundles.

Bundle-Item Mapping

We visualize the bundle-item mapping using Gephi (Figure 4). We notice that the current bundles in the Steam Store are mostly generated in two ways. First is by grouping the game series or assets (soundtracks, skins, books) together. For example, in the top right corner, we can see a bundle (in red circle) that contains the three character skins for 'War for the Overworld'. The other interesting pattern is that sometimes bundles are generated by grouping games of similar themes together. For instance, the bundle (in red circle) on the bottom contains three games of the same theme - school life role playing.
In Figure 5, we plot the distribution of the number of bundles a game appears in as well as the number of games a bundle contains. We see that this graph, by nature, is very scattered in that it contains a

[1] `https://cseweb.ucsd.edu/~jmcauley/datasets.html`

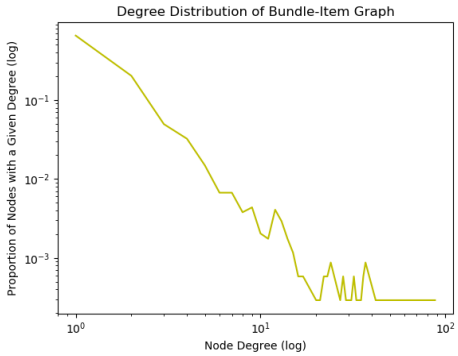

Figure 3: Node degree distribution in the bundle-item graph.

lot of disjoint components. Number of weakly connected components is 412, and the sizes of these components are small (Figure 6), most of which are smaller than 15.

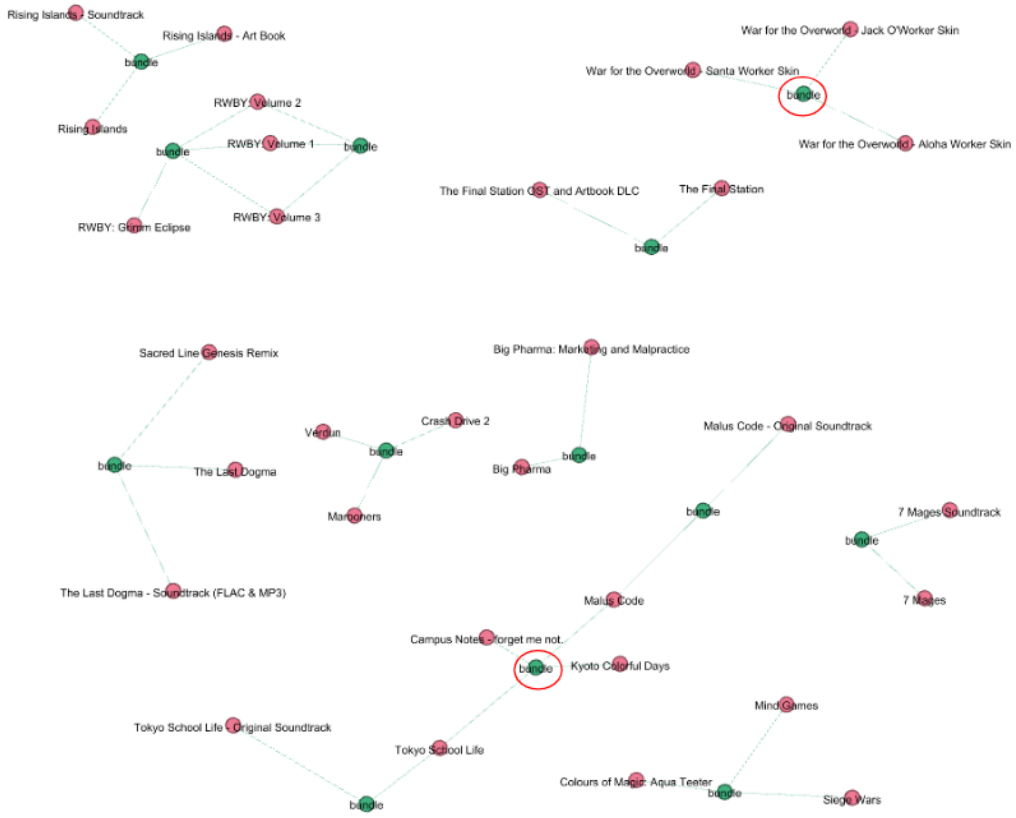

Figure 4: A subset of the bundle-item mappings visualized with Gephi. The green nodes are the bundles and the red nodes are the individual games.

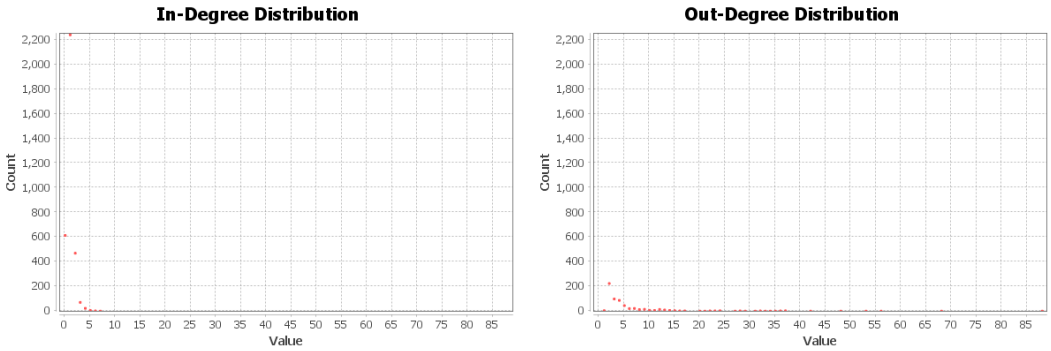

Figure 5: In-degree distribution shows the distribution of the number of bundles a game belongs to. Out-degree distribution shows the distribution of the number of games that a bundle has.

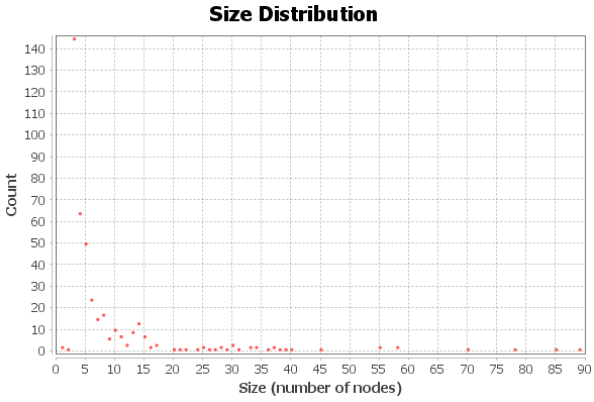

Figure 6: Distribution of the size of WCCs in the graph. We see that the majority of the WCCs have small sizes, which is consistent to the degree distributions in Figure 5 since most games do not appear in more than 3 bundles. Note that there are no SCCs in the graph as the edges are directed (from bundle to games).

5 Task: Bundle Preference Ranking

5.1 Algorithm: Bayesian Personalized Ranking (BPR)

In (3), the authors introduce the Steam dataset we will be using and two algorithms relevant to bundle recommendation:

- Bayesian Personalized Ranking (BPR) to rank the items bundles.
- A greedy algorithm that uses the learned parameters to generate new bundles.

The goal of BPR is to derive a personlized ranking $>_u$ over items (or bundles). Formally, we want an estimator $\hat{x} : U \times I \to \mathbb{R}$ encoding the compatibility between a user and an item (or bundle). We want

$$i_p >_u i_n \leftrightarrow \hat{x}_{u,i_p} >_{\mathbb{R}} \hat{x}_{u,i_n} \tag{1}$$

where left hand side means that item p is more desirable than item n for user u. Right hand side means that the estimator is able to preserve this relationship. Hence, we want to come up with an estimator that violates Eq (1) as little as possible. We will achieve this by solving an optimization problem **BPR**. The objective value we are minimizing is

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

where, σ is the sigmoid function, θ is the parameter vector of the compatibility function. λ is the regularization hyper-paramter. $\hat{x}_{u,p}$ and $\hat{x}_{u,n}$ represent compatibility estimates that the uesr u would

purchse item p and n respectively. i_p and i_n are a positive and negative item, so $\sigma(\hat{x}_{u,i_p}(\theta) - \hat{x}_{u,i_n}(\theta))$ is like the embedding difference term in the TransE objective. D represents the training set. Each entry in D consists of the user, a positive item (or bundle), a negative item (or bundle). Positive means being bought by user. Negative is the opposite.
The definitions for $\hat{x}_{u,i}$ for item and $\hat{x}_{u,b}$ for bundle are:

- $\hat{x}_{u,i} = \beta_i + P_u \cdot Q_i$, where β_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 optimzing BPROpt over D_{item}.
- $\hat{x}_{u,b} = \frac{1}{|B_b|} \sum_{i \in B_b} [\kappa\beta_i + (\mu P_u)(\omega Q_i)] + C_{c_b} + N_b$, where β, P, are learned from the item BPR model. μ, ω are $k \times k$ dimensional matrix adjustment parameters for P and Q respectively. c_b is 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 $\max_{j \in B} |B_j|$ 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.

We use AUC to evaluate bundle BPR.

$$AUC = \frac{1}{|T|} \sum_{(u,p,n) \in T} \delta(\hat{x}_{u,i_p} - \hat{x}_{u,i_n} > 0) \tag{3}$$

δ is the indicator function and T is the fraction of the data withheld for testing.

We ran the Bundle BPR algorithm for generating the bundle preference ranking. The AUC for that is 0.898, which means the bundle preference estimator gives correct preference ordering on 89.8 percent of the test tuples.

6 Task: Bundle Prediction

6.1 Algorithm: Coordinated Matrix Minimization

In (4), the authors proposed an EM algorithm for predicting hyperlinks. We describe it in the context of our problem. We have a hypernetwork $S : \mathbb{R}^{n \times m}$, where n is the number of nodes (users and games) and m is the number of hyperlinks. The i_{th} column in S have 1's on the nodes that are in the hyperlink, and 0 elsewhere. Then, $A = SS^T$ will be the adjacency matrix representation of the network where A_{ij} is the co-occurrence count of vertex i, j in all hyperlinks.
Out training set on the hypernetwork is missing some hyperlinks ΔS - these are the ones we want to predict). We have m' number of candidate hyperlinks generated from the distribution of the training set, and we want to find the ones to add to the training set. We will use the columns of $U : \mathbb{R}^{n \times m'}$ to store the m' candidate hyperlinks and a diagonal matrix : $\{0, 1\}^{m' \times m'}$ to indicate which candidate hyperlinks to keep. Ideally, we want $\Delta A = U\Lambda U^T$ so that $A + U\Lambda U^T \approx WW^T$.
We use an EM algorithm to find the completed hypernetwork W.

E-step Fix W, minimize $||A - WW^T + U\Lambda U^T||_F^2$ over Λ s.t. $_i \in \{0, 1\}$, $i = 1, ..., m'$.

M-step Find W by minimizing $||A - WW^T + U\Lambda U^T||_F^2$ over W s.t. $W \geq 0$.

The result of this algorithm is the completed network W with the hyperlinks in the training set and the additional proposed hyperlinks. We will evaluate the model by looking at the fraction of the real bundle purchases in those proposed hyperlinks.
We ran the CMM on the Steam game dataset, and the accuracy is 0.486, which means that of all the proposed bundles, 48.6% of them will be purchased by users.

6.2 Algorithm: Graph Convolutional Networks

In CMM, we are simply leveraging the distribution of the items/bundles and the purchases. Ideally, we want to be able to use the meta data for this dataset, as they can be informative. For instance, the reviews of the items in the bundles can infer how likely a user is going to purchase that bundle.

For this experiment, we will perform hyperlink prediction using GraphSage (1) so we can incorporate the meta data as well as obtain a comparison between CMM.
We formulate the problem of hyperlink prediction as a node classification on the dual of the hypergraph (2). Suppose our hypernet is $H = (V, E)$ where $V = \{v_1, ..., v_n\}$ is the set of nodes (users and items). $E = \{e_1, .., e_m\}$ is the set of hyperlinks. The non-zero entries in $e_i \in \{0, 1\}^{|V|}$ represent a purchasing record involving the user node and the game nodes correponding to those entries. The dual of H is $H^* = (V^*, E^*)$ where $V^* = E$. $E^* = \{e_1^*, ..., e_n^*\}$ where $e_i^* = \{e \in E : v_i \in e\}$. Essential, an edge in the dual graph means that the two end nodes (two bundles) share some item(s) in common. For a visual example, see Figure 7. Note that the hyperlink prediction on H is equivalent

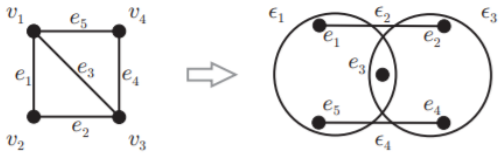

Figure 7: Example of hypergraph duality from (2). A hypergraph $G = (V, E)$, where $V = \{v_1, v_2, v_3, v_4\}$ and $E = \{e_1, e_2, e_3, e_4, e_5\}$ with its dual hypergraph $G^* = (V^*, E^*)$, where $V^* = \{e_1, e_2, e_3, e_4, e_5\}$ and $E^* = \{\epsilon_1, \epsilon_2, \epsilon_3, \epsilon_4\}$ such that $\epsilon_1 = \{e_1, e_3, e_5\}$, $\epsilon_2 = \{e_1, e_2\}$, $\epsilon_3 = \{e_2, e_3, e_4\}$ and $\epsilon_4 = \{e_4, e_5\}$.

to binary node classification on H^*. We can use GraphSage to do that. The only problem now is that the edges in H^* are still hyperedges. We will fix that by replacing each hyperedge among k nodes by a clique of size k. We claim that this simplification does not break our assumption made for node embeddings - two nodes have similar embeddings if they are in the same clique. We summarize our algorithm as below.

- Get the vocabulary V of all the words that have appeared in the game reviews and user reviews. All words with document frequency less than 10 were removed.
- In addition to the observed hyperlinks E in H, we randomly sample an additional N number of hyperlinks where each one of them include 1 user node and several game nodes.
- The observed hyperlinks have positive labels. The labels of the sampled hyperlinks are the ones we want to predict.
- Generate attributes for each node. The attributes we used are: the number of reviews the game got, the sentiment (a 10-class categorical variable) of the game review, etc. The size of the feature vector is 13.
- Generate hypergraph H. When we turn $e_i = \{v_j : v_j \in e_i\}$ into a node v_i^* in H^*, we get the attribute for v_i^* by concatenating the attributes of the nodes in e_i and dividing by the number of nodes in the bundle.
- Run GCN on H^* by minimizing the fraction of misclassifying the observed labels.
- Predict the labels for the sampled hyperlinks. If the predicted label for $v_i^* = e_i$ is positive, then we say that the user in e_i will purchase the bundle consists of items in e_i.

To compare to the CMM algorithm, we will evaluate the model by the fraction of true positives, which can be interpreted as the fraction of actual purchases in the proposed hyperlinks. After some hyperparameter tuning, with the same number of randomly sampled negative bundles, we are able to achieve 90.1 percent accuracy on the test set using GCN (with 5 layers, 0.0001 learning rate, and 1000 epochs).

7 Discussion and Future Work

To summarize, we have tried the Bundle BPR algorithm for generating the bundle preference ranking. The AUC for that is 0.898, which means the bundle preference estimator gives correct preference ordering on 89.8 percent of the test tuples. Although this sounds impressive, the Bundle BPR has an intrinsic limitation, which is that it only learns preference estimators for existing users and existing items in the dataset. It cannot generalize to new users or new items.

We also ran CMM for hyperlink prediction and formulated an alternative algorithm using GraphSage. The accuracy for CMM is 0.486, which means that of all the proposed bundles, 48.6% of them will be purchased by users.

The third algorithm we proposed using GCN, which does much better than CMM with a 90.1% accuracy.

As for future work, we can make this work more applicable in real e-commerce settings. For example, in our analysis, we formulated the problem as a binary node classification problem where each node represent a bundle. If a node is positive, then it means this bundle is in fact a bundle in the dataset and there are at least one customer who purchased this bundle. To have more fine-grained analysis and derive more interesting business insights, we may want to make it a multi-class classification problem, where each class shows a desirability level of the bundle. For example, we can have different classes for bundles that are purchased by 100+, 1000+, and 10000+ people. Note that for this dataset, we were unable to do so, because the distribution of the user purchases are extremely skewed, so making it a multi-class prediction problem would be sub-optimal.

References

[1] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017.

[2] J. Lugo-Martinez and P. Radivojac, "Classification in biological networks with hypergraphlet kernels," *arXiv preprint arXiv:1703.04823*, 2017.

[3] A. Pathak, K. Gupta, and J. McAuley, "Generating and personalizing bundle recommendations on steam," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2017, pp. 1073–1076.

[4] M. Zhang, Z. Cui, S. Jiang, and Y. Chen, "Beyond link prediction: Predicting hyperlinks in adjacency space," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[5] D. Zhou, J. Huang, and B. Schölkopf, "Learning with hypergraphs: Clustering, classification, and embedding," in *Advances in neural information processing systems*, 2007, pp. 1601–1608.

[6] T. Zhu, P. Harrington, J. Li, and L. Tang, "Bundle recommendation in ecommerce," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 2014, pp. 657–666.