Multi-behavior Recommendation with Graph Convolutional Networks

Bowen Jin[1], Chen Gao[1], Xiangnan He[2], Depeng Jin[1], Yong Li[1],
[1]Beijing National Research Center for Information Science and Technology (BNRist),
Department of Electronic Engineering, Tsinghua University
[2]School of Information Science and Technology, University of Science and Technology of China
liyong07@tsinghua.edu.cn

ABSTRACT

Traditional recommendation models that usually utilize only one type of user-item interaction are faced with serious data sparsity or cold start issues. Multi-behavior recommendation taking use of multiple types of user-item interactions, such as clicks and favourites, can serve as an effective solution. Early efforts towards multi-behavior recommendation fail to capture behaviors' different influence strength on target behavior. They also ignore behaviors' semantics which is implied in multi-behavior data. Both of these two limitations make the data not fully exploited for improving the recommendation performance on the target behavior.

In this work, we approach this problem by innovatively constructing a unified graph to represent multi-behavior data and proposing a new model named MBGCN (short for ***M**ulti-**B**ehavior **G**raph **C**onvolutional **N**etwork*). Learning behavior strength by user-item propagation layer and capturing behavior semantics by item-item propagation layer, MBGCN can well address the limitations of existing works. Empirical results on two real-world datasets verify the effectiveness of our model in exploiting multi-behavior data. Our model outperforms the best baseline by 25.02% and 6.51% averagely on two datasets. Further studies on cold-start users confirm the practicability of our proposed model.

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

Multi-behavior-Recommendation, Collaborative Filtering, Graph Convolutional Networks

ACM Reference Format:
Bowen Jin[1], Chen Gao[1], Xiangnan He[2], Depeng Jin[1], Yong Li[1], . 2020. Multi-behavior Recommendation with Graph Convolutional Networks. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401072

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401072

1 INTRODUCTION

Personalized recommender system has become a widely used service to alleviate the issue of information overload nowadays [28]. Collaborative filtering (CF) [29], the most extensively accepted paradigm for building a recommendation model, can learn user interest and estimate preference from the collected user behavioral data, *i.e.*, historical feedback such as purchase. Traditional CF models [17, 20, 21, 23, 27] are designed for a single type of behavior, which is directly relevant to platform profit in most cases, such as purchase behavior in e-commerce platform; However, in real-world applications, this may lead to serious cold-start or data sparsity issue. For example, on an e-commerce website, a CF model built with only purchase behavior can hardly achieve good recommendations for a new user without historical purchase. But the good news is that the platforms can collect some other types of behaviors that easily happen, such as click and browse. In other words, recommender systems should have the ability to make use of other types of behaviors, *auxiliary behaviors*, to help predict users' future interaction on the *target behavior*, which is the multi-behavior recommendation.

Existing researches [3, 7, 10, 22, 26, 31, 33] approach this task from two aspects. The first category utilizes multi-behavior data into the sampling process and builds multi-sampling pairs to reinforce the model learning process [22, 26, 27]. For example, MCBPR [22] assumes that there exists an importance order between behaviors, and it extends BPR [27] by building sampling pairs with a type of positive behavior and another type of weaker behavior. This is further extended by [26], designing a more complex training-pair sampling method based on multi-behavior data. The second category tries to design model to capture multi-behavior information [3, 10, 31, 33]. For instance, matrix factorization based models [31] conduct the factorization on multiple behavior matrices at the same time; [33] designs a multi-objective optimization method. [3, 10] respectively propose a deep model for multi-task learning and assume an artificially given strength order between behaviors.

Despite effectiveness, these works suffer from two limitations:

- **The *strength* of multiple types of behaviors is not sufficiently utilized.** On one hand, ignoring to model the strength of behaviors will make some useful signals lost. For example, the adding-to-cart behavior is obviously a stronger signal compared with the click behavior; thus, accurately modeling the close relation between adding-to-cart and purchase can improve the performance. On the other hand, it is not reasonable to roughly regard all auxiliary behaviors as weaker ones comparing with

*The first two authors contributed equally to this research.

the target behavior. For example, on the e-book recommender system, sharing a book to friends reflect a stronger user preference compared with purchasing a book, which is the target behavior. In short, the multi-feedback recommendation model is required to reveal and further utilize the various strength of different behavior types from the data.

- **The *semantics* of multiple types of behaviors are not considered.** The semantics of behaviors can be understood as the meaning of a type of behavior or, in other words, the reason why a user-item interaction happens under a specific type of behavior. From another perspective, there should be some common characteristics or special relations among the interacted items under each type of behavior. For example, when users are clicking and viewing products on an e-commerce websites, the co-clicked/viewed products may be replaceable (such as iPhone and Google Pixel); when it comes to purchases, those co-purchased products may be complementary (such as iPhone and AirPods). The item-to-item relation can serve as the solution to reveal the semantics for collaborative recommendation tasks when we have no side information of items, such as category or brand.

In short, the limitations of existing methods lie in the fact that they cannot thoroughly address the above two challenges: modeling *user-to-item based strength* and *item-to-item based semantics* of multiple types of behaviors.

To address them, we propose to construct a unified heterogeneous graph based on multiple types of behavioral data. With user/item represented as nodes and different types of behaviors represented as multiple types of edges of the graph, the problem of modeling user-to-item based strength and item-to-item based semantics turns to model heterogeneous edges and item-user-item meta-path. We further propose a solution named ***M**ulti-**B**ehavior **G**raph **C**onvolutional **N**etwork* (MBGCN) to take advantage of the strong power of graph neural networks in learning from complicated edges and high-order connectivity on graph for addressing above two challenges.

To be more specific, we construct a heterogeneous graph consisting of two kinds of nodes (users and items) and multiple types of edges, where an edge connecting user node and item node refers to a specific type of user-item feedback. First, such graph-structured interaction data do not set any prior constraint or assumption on the preference strength. In order to capture the various strength of different behaviors, we propose behavior-aware item-to-user propagation layers for each behavior to propagate neighboring item nodes' embedding to the user node. This makes the model able to distinguish the different strengths of different auxiliary behaviors. Secondly, we design item-to-item propagation layers operating on item nodes' embeddings and co-interacted neighboring (second-order) items, which helps to capture different CF semantics of item similarity for various behaviors and enhance the learning for item embeddings. Through these designs, our MBGCN method effectively addresses the main challenges and helps to exploit auxiliary behaviors for a better recommendation on the target behavior.

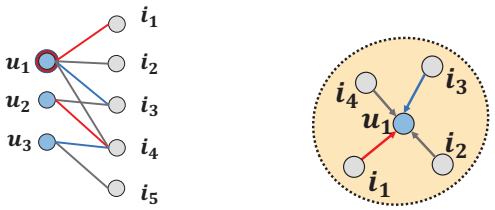
To summarize, the main contributions of this work are as follows:

- We propose to construct a heterogeneous graph for representing multi-feedback data without any constraint on the preference strength of each type of behavior.
- For the constructed graph, we propose a graph convolutional network based model for recommendation. On the one hand, we design behavior-aware user-to-item embedding propagation layers to capture the diverse influence of different behaviors; on the other hand, we design item-to-item embedding propagation layers for modeling item-to-item similarity which reflects the various semantics of different behavior.
- We conduct extensive experiments on two real-world datasets. Experimental results show that our model can effectively improve the recommendation performance by 25.02% and 6.51%, respectively, comparing to the best performance baseline. Further studies on cold-start users validate the high application value of our model.

The remainder of the paper is as follows: First we formalize the problem in Section 2 and then present our proposed method in Section 3. After conducting experiments in Section 4, we review the related work in Section 5. Last, we conclude the paper in Section 6.

2 PROBLEM FORMULATION

In real-world scenarios of the online information system, users can interact with items provided by the platform in multiple manners, click, collect, purchase, share, etc. Among various types of user-item interactions, there is always one type that directly determines the platform's profit. Recommender systems are always designed towards that single type of behavior. For example, e-commerce recommender systems are always designed for purchase behavior, and App recommender system are designed for download behavior. As mentioned in the Introduction, due to the data sparsity and cold-start issue, the single-behavior recommendation may achieve poor performance. In this work, we aim to design a recommendation model for the target behavior by exploiting other types of feedback.



(a) U-I Interaction Graph (b) Local Graph of u_1

Figure 1: An illustration of the user-item multi-behavior graph, where node u_1 is the target user to provide recommendations for.

Assume that the number of behaviors is T, and interaction matrice Y^t denotes if user has interacted with item under behavior t or not. All behavior matrices can be expressed as $\{Y^1, Y^2, ..., Y^T\}$, among which $\{Y^1, Y^2, ..., Y^{T-1}\}$ are represented for auxiliary behaviors, and Y^T is expressed for the target behavior. Specifically, interaction matrice Y^t is in the binary form, of which each entry has value 1 or 0, defined as follows for user u and item i,

$$y_{ui}^t = \begin{cases} 1, & \text{if } u \text{ has interacted with } i \text{ under behavior } t; \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

Note that there is no constraint on temporal order or strength order of behaviors. In other words, behavior $t-1$ does not have to happen before t, and behavior $y_{ui}^{t-1}=1$ does not have to reflect weaker or stronger user preference compared with $y_{ui}^{t}=1$. Then the task of multi-behavior recommendation can be formulated as:

Input: The user-item interaction data of T types of behaviors, $\{Y^1, Y^2, ..., Y^T\}$.

Output: A recommendation model that estimates the probability that a user u will interact with an item i under the T-th behavior, i.e. target behavior.

3 METHODOLOGY

We now come to the details of our method, the architecture of which is presented in Figure 2. Our model has four important components: 1) a shared layer which provides initialization for the user and item embedding; 2) a user-item propagation layer to learn the strength of each behavior and extract collaborative filtering signal based on multi-behavior user-item interaction at the same time; 3) an item-item propagation layer to refine items' special relation or in other words, behavior semantics based on types of behavior; 4) a joint predicting module.

3.1 Unified Heterogeneous Graph

We aim to use all kinds of behaviors to perform the recommendation for the target user. Therefore, a unified heterogeneous graph is built to model the investigated problem. The input interaction data is represented by an undirected graph $\mathbf{G}=(\mathbf{V},\mathbf{E})$, where nodes are $\mathbf{V}$ consisting of user nodes $u\in\mathbf{U}$ and item nodes $i\in\mathbf{I}$. The edges in $\mathbf{E}$ contain different user-item interaction edges of different behavior, namely $(u,i)_t, t\in N_r$, where N_r is the set of all behavior types that occur between user and item. There will be an edge $(u,i)_t$ built, when $y_{ui}^t=1$. We use different kinds of edges to represent different behavior so that behavior-based information between user and item can be extracted. Besides, some meta-paths are built between items based on users' co-behavior to extract item relevance better. For example, if many users buy iPhone and AirPods at the same time, there will be an item-purchase-user-purchase-item meta-path between iPhone node and AirPods node. Since meta-paths are built based on users' co-behavior, the number of meta-path types equals the number of behavior types.

3.2 Shared Embedding Layer

Similar to the existing graph models in [9, 31, 42, 43], we use embedding vector $p_i^{(0)}\in\mathcal{R}^d$ and $q_j^{(0)}\in\mathcal{R}^d$, where d is embedding size, to describe a user and an item. User embedding vectors and item embedding vectors can be expressed by embedding matrix $\mathbf{P}$ and $\mathbf{Q}$ respectively,

$$\boldsymbol{P}=\{p_{u_1}^{(0)},p_{u_2}^{(0)},...,p_{u_n}^{(0)}\},\quad \boldsymbol{Q}=\{q_{i_1}^{(0)},q_{i_2}^{(0)},...,q_{i_m}^{(0)}\}.$$

To ensure the extensibility of our method, one hot vector is used as input to describe the ID of a user or an item. We use matrix multiplication here to obtain the embedding for such user (item) with one hot vector as follows,

$$p_{u_k}^{(0)}=\boldsymbol{P}\cdot ID_k^U,\quad q_{i_j}^{(0)}=\boldsymbol{Q}\cdot ID_j^V.$$

where ID_k^U and ID_j^V are one-hot vector for user u_k and item i_j respectively. It is worth noting that embeddings in matrix $\mathbf{P}$ and $\mathbf{Q}$ are served as the initialization feature of users and items, which can be seen as the input feature for each user and item in the framework of graph neural network [19].

3.3 Behavior-aware User-Item Propagation

In order to capture the CF signal based on multi-behaviors, we build upon a message-passing architecture between user and item. With embedding propagation, for each node, information from its neighbors will be fused into the embedding, which can reinforce embedding learning and improve predicting effectiveness. When coming to our task, we design a particular user-item propagation method for better utilization of multi-behavior information. Our user-item propagation architecture is made up of two components shown in Figure 3 and Figure 4: an item-to-user embedding propagation module, which learns the importance of each behavior automatically and assigns different weight to items interacted under different behavior when aggregating for user; and a user-to-item embedding propagation module, which aggregates neighbour user information for the item.

3.3.1 ***User Embedding Propagation.*** Our main idea is to consider items' influence on user preference differently according to behavior type by two key factors: behavior inherent strength and interaction sparsity. In terms of inherent strength, it is intuitive that different behaviors have different contributions to the target behavior. However, we argue that the importance of each behavior or, in other words, the contribution of each behavior to the target behavior cannot be measured artificially and should be learned by the model itself. Also, data sparsity should be taken into consideration at the same time. The fact is that although target behavior is of the highest significance among all the behaviors, it is hard to mine a user's interest only based on target behavior when there are few target behavior interactions between the particular user and all items. As a result, other behaviors may play a more important role when we have a few target behavior. Based on these two key factors, we design a behavior-sparsity based item-to-user embedding propagation mechanism as follows.

User Behavior Propagation Weight Calculation. Since different behavior contributes differently to the target behavior, we assign a weight for each behavior, namely w_t for behavior t. In order to fuse behavior importance and behavior sparsity together, we define propagation weight for a particular behavior t for user u denoted as α_{ut} as follows,

$$\alpha_{ut}=\frac{w_t\cdot n_{ut}}{\sum_{m\in N_r} w_m\cdot n_{um}},\tag{2}$$

where w_t is a behavior-wised importance weight of behavior t which is the same for all users, and n_{ut} is the count of behavior t operated by user u which is different depending on user. To be specific, behavior with larger w will be of higher importance comparing to behavior with smaller w. $\sum_{m\in N_r} n_{um}$ is the total interaction of user u. α_{ut} is the final propagation weight of behavior t for user u which will be used in propagation layer and $\sum_{t\in N_r}\alpha_{ut}=1$. Note that w_t is learned by the model so that the importance of each

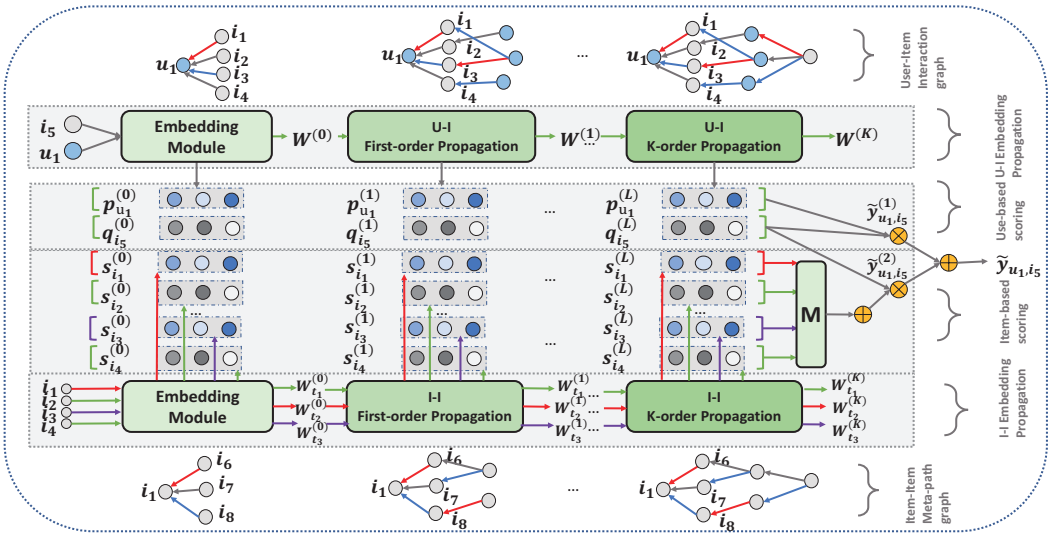
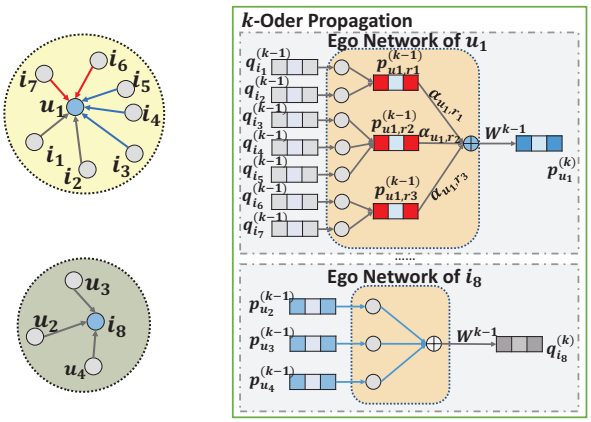


Figure 2: The illustration of MBGCN model, where node u_1 is the target user and i_5 is the target item.



(a) Local Graph (b) k-order Propagation Process

Figure 3: behavior-aware user-item propagation.

behavior can be learned automatically, without having us assigning importance to each behavior.

Neighbour Item Aggregation Based on behavior. For each user, different behavior contributes differently to the target behavior, but it is intuitive that items that are interacted under the same behavior reflect user's similar preference strength. As a result, items that have the same behavior interaction with user are aggregated together so as to obtain one embedding for each behavior, namely $p_{u,t}^{(l)}$ for user u under behavior t, which is defined as follows,

$$p_{u,t}^{(l)} = \text{aggregate}(q_i^{(l)} | i \in N_t^I(u)),$$

where l refers to the l-th layer, $N_t^I(u)$ is the set of items that user u has interacted under behavior t, $q_i^{(l)}$ is item embedding of item i on l-th layer and $p_{u,t}^{(l)}$ is the item aggregated embedding for user u under behavior t on l-th layer. Note that aggregation function can be a function such as simple mean function, mean function with sampling, max pooling and so on. We use simple mean function here and leave other function for future exploring.

Behavior-level Item Propagation for User. We sum neighbor item aggregation embedding together according to weight α_{ut} and then go through an encoder matrix to obtain the final neighbor item aggregation for user u. Similar to work [38], we use a graph neural network without activate function here in order to refine information based on multi-behavior as follows,

$$p_u^{(l+1)} = W^{(l)} \cdot (\sum_{t \in N_r} \alpha_{ut} p_{u,t}^{(l)}),$$

where $p_u^{(l+1)}$ is the embedding of user u in the $(l+1)$-th layer and $W^{(l)}$ is the encoding matrix for information aggregation in the l-th layer. It is worth mentioning that different behavior has a different contribution to user embedding based on coefficient $\alpha_{u,r}$, which depends on both behavior significance and user interaction quantity under each behavior.

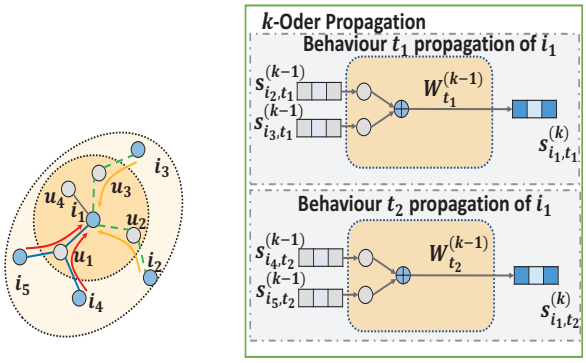
3.3.2 ***Item Embedding Propagation.*** In the item-to-user embedding propagation layer above, we assign different weights for different behavior aggregation embedding based on the fact that behaviors are subjectively conducted by users so that items interacted under different behavior should reflect different user feature. However, the case is not the same to user-to-item propagation, since the feature of the item is static. Regardless of different kinds of behaviors, we assume that different user has the same contribution to one item. Hence, a user-to-item propagation method is shown below, aggregating user embedding $p_j^{(l)}$ for the next layer item embedding $q_i^{(l+1)}$ as follows,

$$q_i^{(l+1)} = W^{(l)} \cdot \text{aggregate}(p_j^{(l)} | j \in N^U(i)),$$

where $N^U(i)$ is the set of user that item i has interacted with, $p_j^{(l)}$ is the embedding for user j on l-th layer and $W^{(l)}$ is the encoding matrix for information aggregation on the l-th layer. Although behavior types are not considered here in user-to-item propagation, it is inappropriate to say that multi-behavior can not be used in item feature learning. In fact, item relevance or in other words, behavior semantics can be learned from multi-behavior data from item-to-item perspective which will be shown in Section 3.4.

3.4 Item-Relevance Aware Item-Item Propagation

As discussed above, item information reflected in multi-behavior is not reasonable to be utilized from the user-to-item propagation perspective; we need to design a better method for item information extracting. What is ignored in previous work [3, 10] approaching multi-behavior recommendation is the semantic of different behavior or, in other words, the relevance of items reflected in behaviors. Items that are co-interacted by user may have special connection. Therefore, it is reasonable to diffuse an item's information to items co-interacted according to behavior type.



(a) Local Graph (b) k-order Propagation Process

Figure 4: behavior-aware Item-Item Propagation.

According to discussion above, we design an item-to-item propagation layer based on behavior to seize behavior semantics. We aggregate the embedding $s_{jt}^{(l)}$ of items j that are co-behaviored under behavior t with item i to obtain next layer embedding $s_{it}^{(l+1)}$. Through this mechanism, features of items which are co-behaved with item i will be aggregate to i's embedding, which contributes to better embedding learning and final predicting.

$$s_{it}^{(l+1)} = W_t^{(l)} \cdot \text{aggregate}(s_{jt}^{(l)} | j \in N_t^I(i)),$$

where $N_t^I(i)$ is the set of items which are interacted together with item i by users under behavior t and $W_t^{(l)}$ is an encoding matrix for behavior t which can help aggregate information in l-th layer. We have $s_{it}^{(0)} = q_i^{(0)}$ for all $t \in N_r$. After this process, each item will have $|N_r|$ propagation embeddings.

3.5 Joint Prediction

After propagating through L layers, we obtain multiple representations, namely $\{p_u^{(0)}, ..., p_u^{(L)}\}$ for user u and namely $\{q_i^{(0)}, ..., q_i^{(L)}\}$ and $\{s_{it}^{(0)}, ..., s_{it}^{(L)}\}$, $t \in N_r$ for item i. Representations obtained in different layer emphasize messages received from different degree neighbour. To be more specific, $p_u^{(0)}$ contains information of user u itself, $p_u^{(1)}$ contains information of items which are interacted by user u, namely one-degree neighbour and $p_u^{(k)}$ contains information of user u's k-degree neighbour. Case is the same for q_i and s_{it}. For final prediction, information from different kinds of neighbour are all important. As a result, we concatenate them together to get the final embedding for user and item:

$$\begin{aligned} p_u^* &= p_u^{(0)} || ... || p_u^{(L)}, \\ q_i^* &= q_i^{(0)} || ... || q_i^{(L)}, \\ s_{it}^* &= s_{it}^{(0)} || ... || s_{it}^{(L)}, t \in N_r, \end{aligned}$$

where $||$ is the concatenation operation. By doing so, we can not only enrich the final embedding used for prediction with information from different layers but also allow controlling of the range of propagation by adjusting L.

Since user-item propagation layer learns user embedding p_u^* and item embedding q_i^* from user-item interaction directly, we can calculate a user-based score with p_u^* and q_i^* [27, 37], which is called user-based score. Furthermore, s_{it}^* learned from item-to-item propagation contains information of item relevance, which can be used to calculate a score of target item together with items the user has interacted with, namely item-based score.

3.5.1 ***User-based CF Scoring.*** We employ simple interaction function of inner product here in order to calculate the score based on user-item propagation, as follows,

$$y_1(u, i) = p_u^{*T} \cdot q_i^*$$

It's worth noting that user embedding utilized here contains auto-learned behavior strength information.

3.5.2 ***Item-based CF Scoring.*** As mentioned above, there may exist behavior semantics in multi-behavior recommendation, that user's co-behavior toward two items may imply relevance between these two items, and the relevance between items may influence user's behavior (purchase or not). Therefore, a scoring mechanism is deployed here so as to mine behavior semantics and calculate relevance score $y_2(u, i)$ between the target item i and items user u has interacted with. Item relevance scores calculated from different behavior are summed up to obtain the final relevance score,

$$y_2(u, i) = \sum_{t \in N_r} \sum_{j \in N_t^I(u)} \frac{s_{jt}^{*T} \cdot M_t \cdot s_{it}^*}{|N_t^I(u)|},$$

where $N_t^I(u)$ is the set of items that has been interacted by user u under behavior t and $M_t \in R^{d' \times d'}$ with $d' = d \times (L+1)$ is a trainable matrix which measures relevance between two item embeddings based on behavior t.

3.5.3 ***Combined Scoring.*** User-based CF score and Item-based CF score are combined together with a hyperparameter λ, which

Table 1: Dataset statistics.

Dataset	Users	Items	purchase	cart	collect	click
Tmall	41,738	11,953	255,586	1,996	221,514	1,813,498
Beibei	21,716	7,977	304,576	642,622	—	2,412,586

adjusts the weight of user-based scoring and item-based scoring in final score as follows,

$$y(u,i) = \lambda \cdot y_1(u,i) + (1-\lambda) \cdot y_2(u,i).$$

By summing the two scores up, the final score not only extracts the collaborative filtering signal between user and item but also considers behavior semantics.

3.6 Model Training

To learn the parameters, we optimize the model with BPR loss, which is widely used in the recommendation system [27, 37, 43]. It emphasizes the relative order between observed and unobserved user-item interaction and claims that observed interaction, which is instructional for user's preference learning, should be assigned a higher prediction score than unobserved ones.

Thus, we can define the loss function as follows,

$$Loss = \sum_{(u,i,j)\in O} -ln\sigma(y(u,i) - y(u,j)) + \beta \cdot ||\Theta||^2,$$

where $O = \{(u,i,j)|(u,i) \in R^+, (u,j) \in R^-\}$ denotes the set of pairwise target behavior training data, R^+ represents observed target behavior and R^- represents unobserved target behavior; $\sigma(\cdot)$ is the sigmoid function, Θ denotes all trainable parameters and β is the L2 normalization coefficient which controls the strength of the L2 normalization to prevent overfitting.

Although today's deep learning models have strong representation ability, they usually suffer from overfitting problems. Following recent work [2, 37], we propose to employ two widely used dropout methods: message dropout and node dropout to some user nodes and item nodes. Message dropout randomly drops out some flowing message with a possibility p, while node dropout randomly deletes a particular node and withdraws all its flowing information.

4 EXPERIMENT

In this section, we conduct experiments on two real-world e-commerce datasets to evaluate our proposed MBGCN method. Our purpose is to answer the following four research questions.

- **RQ1:** Does introducing multi-behavior data improve recommendation performance? How does our method perform comparing to the state-of-the-art models that aim to learn from multi-behavior data?
- **RQ2:** How do different settings of item-to-user propagation weight and item-to-item propagation method influence our model's effectiveness?
- **RQ3:** How can MBGCN alleviate the cold-start problem?
- **RQ4:** How do the model hyperparameters (λ, message dropout, and node dropout) affect the final performance of our model?

4.1 Experimental Settings

4.1.1 Dataset. To evaluate the performance of MBGCN, we experiment on two real-world e-commerce datasets: Tmall and Beibei. We summarize the statistics of the two datasets in Table 1.

- **Tmall**[*] This is an open dataset collected from Tmall, the largest e-commerce platform in China. There are 41738 users and 11953 items with four types of behavior: purchasing, carting, collecting, and clicking recorded in this dataset.
- **Beibei**[†] This is the dataset obtained from Beibei, the largest infant product e-commerce platform in China. There are 21716 users and 7977 items with three types of behaviors, including purchasing, carting, and clicking collected in this dataset.

4.1.2 Evaluation Metrics. To evaluate the performance of each model, we use two widely used metrics called Recall and NDCG which are defined as follows,

- **Recall@K** measures the ratio of test items that have been successfully recommended in the top-K ranking list.
- **NDCG@K** assigns higher scores to hits at a higher position in the top-K ranking list, which emphasizes that test items should be ranked as higher as possible.

4.1.3 Baselines. To demonstrate the effectiveness of our MBGCN model, we compare it with several state-of-the-art methods. The baselines are classified into two categories: one-behavior models that only utilize target behavior records, and multi-behavior models that take all kinds of behavior into consideration.

One-behavior Model:

- **MF-BPR**[27] BPR is a widely used method which optimizes pairwise loss with the assumption that observed interaction should have higher score than unobserved ones.
- **NeuMF**[15] This is the state-of-the-art neural CF method which uses GMF and MLP at the same time to capture user-item interaction signal.
- **GraphSAGE-OB**[13] GraphSAGE is one of the most widely used deep graph neural network model which enriches node embedding with its neighbours' information by embedding propagation and aggregation. Here only target behavior is used to build the user-item bipartite graph and we call this model GraphSAGE-OB.
- **NGCF-OB**[37] NGCF is the state-of-the-art graph neural network model which has some special design to fit graph neural network into recommender system. Similar to GraphSAGE-OB, here only target behavior is used to build the user-item bipartite graph.

Multi-behavior Model:

- **NMTR**[10] This is a state-of-the-art deep model for multi-behavior recommendation. The authors assume that there exists a strict importance order between behaviors and propose a sequential deep model that adopts NCF [15] for each type of behavior under a multi-task learning framework. [3] propose a similar model, combining NCF and multi-task learning, with another loss function [16]. However, in our problem, we do not make any prior assumptions about the sequence of behaviors. Thus, we only choose NMTR for comparison. To make it applicable, we follow the specified sequential order in the original paper.

[*] https://tianchi.aliyun.com/dataset/dataDetail?dataId=649
[†] https://www.beibei.com

- **MC-BPR**[22] MC-BPR is one of the first paper that focuses on utilizing multi-behavior data in recommender system. It assumes that different behavior reflects different order preference between user and item, and this can be used as prior knowledge to build more diverse training pairs.
- **GraphSAGE-MB**[13] Different from GraphSAGE-OB, here we use all kinds of behavior to construct the user-item bipartite graph and different behaviors are treated as the same so that there is only one kind of edge in the graph.
- **NGCF-MB**[37] We use the same method with GraphSAGE-MB to build the graph for NGCF-MB. Since all kinds of behavior are used for NGCF, the embedding learned by the model might have stronger representation ability.
- **RGCN**[30] RGCN is the first paper in graph neural network field that considers edge difference. The author designs different propagation layers for different types of edge to capture edge information, which fits our task of multi-behavior recommendation.

4.1.4 Parameters Settings. Our MBGCN is implemented in Pytorch[‡]. The embedding size is fixed to 32 for all models, which is suitable for the model to learn a strong representation embedding [6, 35]. We optimize all models with Adam optimizer [18], having train batch size fixed to 2048. Xavier initialization [11] is used here to initialize the parameters. For all BPR-based methods, we build user-item pairs on-the-fly with eight sampling processes at the same time. When coming to MCBPR, the sampling rate is carefully searched according to the original paper. For NMTR, the loss coefficient is set according to the original paper on two datasets. In terms of hyperparameters, we apply a grid search method here: learning rate is searched in $[1e^{-6}, 3e^{-6}, 1e^{-5}, 3e^{-5}, 1e^{-4}]$, and L2 normalization coefficient is tuned in $[1e^{-7}, 1e^{-6}, 1e^{-5}, 1e^{-4}, 1e^{-3}]$. Besides, we perform node dropout and message dropout for GraphSAGE, NGCF, RGCN, and MBGCN, and pretrain [15] them with embeddings gotten from MF to obtain better performance. Furthermore, we use early stop here to detect over-fitting, and the training process will be stopped if Recall@10 on the validation set does not increase for 40 epochs. We try several layers for MBGCN and find $l = 1$ to be good enough for the two datasets here.

4.2 Overall Performance (RQ1)

We start by comparing the performance of our composed MBGCN with all other baselines. The result on two datasets are reported on Table 2 and Table 3. From the results, we have the following observations.

- **Model Effectiveness.** From both tables, we can find that our MBGCN overperforms all baselines substantially on all Recall@K and NDCG@K metrics. The average improvement of our model to the best baseline is 25.27% and 24.78% for Recall and NDCG on the Tmall dataset and 9.95% and 3.07% on Beibei dataset, which justifies the effectiveness of our model.
- **Graph models require special designs to work well on recommender systems.** At the one-behavior category, NGCF can outperform traditional matrix factorization and simple neural models in most metrics. This demonstrates the strong power of graph convolutional networks since information of the neighbors of users (items) are fused into their embeddings, which empowers the effectiveness of representation learning. However, GraphSAGE and RGCN fail to capture the CF signal on Tmall dataset, since they do not have special designs that make them fit into recommender system. Therefore, efforts in this work to make graph neural networks work well in the multi-behavior recommendation is necessary.
- **Multi-behavior models work well than one-behavior models.** In the comparison of MCBPR, NMTR, MF-BPR, and NCF, we can find that adding multi-behavior information into predicting (MCBPR and NMTR) can improve the performance. The best multi-behavior model can outperform the best one-behavior model on Tmall by 39.28% on Recall and 14% on NDCG, while 14.65% and 11.07% on Beibei, which clarifies the necessity of introducing multi-behavior data.
- **Our model is the best model which can utilize graph structure and multi-behavior information at the same time.** Comparing with the graph neural network models that can extract information from multi-behavior data such as GraphSAGE-MB, NGCF-MB, and RGCN, our MBGCN performs the best. That can be explained in two folds. First, SAGE-MB and NGCF-MB cannot distinguish different behaviors, which may lead to the loss of some of the information in original data; second, RGCN only considers edge types but ignores the contribution strength of each behavior to the target behavior. However, our MBGCN can extract user behavior weight and item relevance from multi-behavior data, which strongly improves the effectiveness. Besides, our MBGCN, with the strong power of graph neural network, can outperform state-of-the-art multi-behavior recommendation model NMTR.

4.3 Ablation Study (RQ2)

4.3.1 Comparison of Different Behavior Weight Design. In order to evaluate the effect of behavior-based user-item propagation, we compare the performance of our model on Tmall dataset with equal α for all behavior, equal w for all behavior and learn-able w in Equation 2, respectively. The result is recorded in Table 4.

It is shown that the model with learn-able w performs better than model with equal w and the model with equal α by 28.82% and 2.38% on Recall and 23.78% and 1.42% on NDCG respectively. This demonstrates that importance weights of behaviors are essential and should be learned by the model itself.

4.3.2 Comparison of Different I2I Propagation Method. Item-to-item propagation is designed here to extract item relevance based on multi-behavior. We do an ablation study to test the effectiveness of item-to-item propagation by comparing the performance of the model with no item-to-item propagation, the model with only target behavior item-to-item propagation and model with all behavior item-to-item propagation. The results are shown in Table 5.

It is shown that model with all behavior item-to-item propagation outperforms model with no item-to-item propagation and model with target behavior item-to-item propagation by 2.81% and 2.91% on Recall and 3.34% and 1.93% on NDCG respectively. This demonstrates that adding behavior based item propagation can help improve performance, especially when all kinds of behavior information are injected into the model.

[‡]https://pytorch.org/

Table 2: Comparisons on Tmall and improvement comparing with the best baseline.

	Method	Recall@10	NDCG@10	Recall@20	NDCG@20	Recall@40	NDCG@40	Recall@80	NDCG@80
One-behavior	MF-BPR	0.02331	0.01306	0.03161	0.01521	0.04239	0.01744	0.05977	0.02049
	NCF	0.02507	0.01472	0.03319	0.01683	0.04502	0.01931	0.06352	0.02252
	GraphSAGE-OB	0.01993	0.01157	0.02521	0.01296	0.03368	0.01474	0.04617	0.01693
	NGCF-OB	0.02608	0.01549	0.03409	0.01757	0.04612	0.02010	0.06415	0.02324
Multi-behavior	MCBPR	0.02299	0.01344	0.03178	0.01558	0.04360	0.01813	0.06190	0.02132
	NMTR	0.02732	0.01445	0.04130	0.01831	0.06391	0.02279	0.09920	0.02891
	GraphSAGE-MB	0.02094	0.01223	0.02805	0.01406	0.03804	0.01616	0.05351	0.01887
	NGCF-MB	***0.03076***	***0.01754***	***0.04196***	***0.02042***	0.05857	***0.02389***	0.08408	0.02833
	RGCN	0.01814	0.00955	0.02627	0.01165	0.03877	0.01426	0.05749	0.01750
	MBGCN	**0.04006**	**0.02088**	**0.05797**	**0.02548**	**0.08348**	**0.03079**	**0.12091**	**0.03730**
	Improvement	30.23%	19.04%	37.04%	24.78%	24.91%	28.88%	8.90%	26.40%

Table 3: Comparisons on Beibei and improvement comparing with the best baseline.

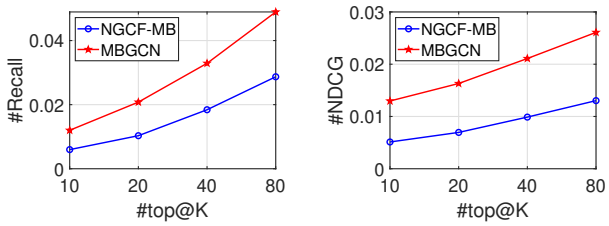
	Method	Recall@10	NDCG@10	Recall@20	NDCG@20	Recall@40	NDCG@40	Recall@80	NDCG@80
One-behavior	MF-BPR	0.03873	0.02286	0.05517	0.02676	0.08984	0.03388	0.14137	0.04258
	NCF	0.04209	0.02394	0.05609	0.02579	0.09118	0.03410	0.15426	0.04022
	GraphSAGE-OB	0.034536	0.01728	***0.06907***	0.02594	***0.11567***	0.03547	0.18626	0.04747
	NGCF-OB	0.04112	0.02199	0.06336	0.02755	0.11051	0.03712	***0.19524***	0.05153
Multi-behavior	MCBPR	0.03914	0.02264	0.04950	0.02525	0.09592	0.03467	0.15422	0.04462
	NMTR	0.03628	0.01901	0.06239	0.02559	0.10683	0.03461	0.18907	0.04855
	GraphSAGE-MB	0.04204	0.02267	0.05862	0.02679	0.09707	0.03451	0.18272	0.04911
	NGCF-MB	***0.04241***	***0.02415***	0.06152	0.02893	0.10370	0.03741	0.01771	0.04987
	RGCN	0.04204	0.02051	0.06354	0.02591	0.09859	0.03309	0.16121	0.04363
	MBGCN	**0.04825**	**0.02446**	**0.07354**	**0.03077**	**0.11926**	**0.04005**	**0.20201**	**0.05409**
	Improvement	13.77%	1.28%	11.76%	3.85%	7.68%	3.30%	6.58%	3.84%

Table 4: Ablation study of user-item propagation weight.

Model	Recall20	NDCG20	Recall40	NDCG40
α_{ur}=**1**	0.04508	0.02068	0.06468	0.02476
Uniform w	0.05586	0.02481	0.08265	0.03075
Learn-able w	0.05797	0.02548	0.08347	0.03079

Table 5: Ablation study of item-item propagation method.

Model	Recall20	NDCG20	Recall40	NDCG40
No propagation	0.05575	0.02451	0.08212	0.02997
Only target	0.05632	0.02458	0.08112	0.03073
All behavior	0.05797	0.02548	0.08347	0.03079



(a) Recall (b) NDCG

Figure 5: Comparison of NGCF-MB and MBGCN solving cold start problem on Tmall.

4.4 Cold-start Problem (RQ3)

When a new user enters recommendation platform, it is hard for most recommender models to recommend item to him since he has no target behavior record, and this problem is called cold-start problem. Different from most models, our model has a item-based scoring module which can assess target item through comparing it with items that the user has interacted with. Owe to this design, we can solve this problem without retraining the model when a new user enters. Since a item relevance calculating module is added into our model, we can predict a score for a test item by calculating the similarity of the particular item with items that user has interacted with, which improves the effectiveness of recommending when user embedding is zero on every dimension.

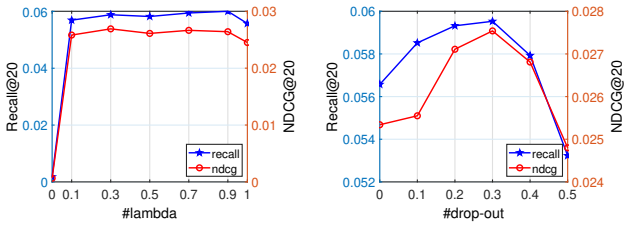
We compare our MBGCN's ability to alleviate the cold start issue with the best baseline NGCF-MB on the Tmall dataset. We randomly choose 1000 users as the cold-start users by deleting their interaction in the training set and then train the model with the remaining users. We follow the same setting with that in section 4.1.4. When these two models reach convergence, we use them to directly perform recommendation to these 1000 cold start users only with their auxiliary-behavior records. The performance of Recall and NDCG is shown in Figure 5.

The result of our MBGCN model outperforms NGCF-MB by 87.96% and 125.37% on Recall and NDCG, respectively. Averagely, the performance of MBGCN under the cold-start setting is 91.43% the performance of MF-BPR under the normal setting, demonstrating the strong power of our MBGCN model.

4.5 Hyper-parameter Study (RQ4)

As our model uses a joint predicting function which fuses user-based score and item-based score together with hyperparameter λ, we start by investigating the impact of λ on the performance first. Moreover, we analyze the influences of node dropout and message dropout.

Firstly, we start experiment with different λ on Tmall dataset to check on its influence. We evaluate λ in $[0, 0.1, 0.3, 0.5, 0.7, 0.9, 1]$ and the result is presented in Figure 6. When $\lambda = 0$, the model only has an item-based scoring component, and it performs bad. With



(a) Study of λ (b) Study of message/node drop-out

Figure 6: Study of λ and dropout ratio on Tmall.

λ increasing from 0.1 to 0.9, MBGCN performs nearly the same. But when $\lambda = 1$, which corresponds to the case when the model only has a user-based CF scoring component, the performance decreases. It can be found that $\lambda = 0.7$ is the best choice for our model. In conclusion, the model with both user-based scoring and item-based scoring performs better than the model with only user-based scoring or only item-based scoring, and user-based scoring is more critical than item-based scoring.

Since we employ message dropout and node dropout techniques here to prevent MBGCN from overfitting, we do experiment on Tmall dataset with ratio selected in $[0, 0.1, 0.3, 0.4, 0.5]$ to explore how message dropout ratio and node dropout ratio (they are assigned the same value) can affect the performance of MBGCN. The result is shown in Figure 6. With the dropout ratio increases from 0 to 0.5, the performance of our model increases and then decreases. The best performance is reached at 0.3. When the dropout ratio is less than 0.3, the model suffers from overfitting, and when the dropout ratio is larger than 0.3, the model degenerates. In conclusion, it is appropriate to choose the message dropout ratio and node dropout ratio around 0.3, which can prevent overfitting without making the model degenerate.

5 RELATED WORK

5.1 Multi-behavior Recommendation

Multi-behavior recommendation [3, 7, 10, 22, 33] utilizes multiple user-item feedback for enhancing recommendation on target behaviors. In general, existing works on multi-behavior recommendation can be divided into two categories.

The first category of works [3, 10, 33] approached this task in a multi-task manner. Specifically, these works considered modeling multiple types of behaviors as multiple tasks and introduced parameter sharing across tasks for joint learning from multi-behavior data. Tang *et al.* [33] proposed to share user embedding matrices among several matrix factorization tasks for each behavioral matrix. Gao *et al.* [10] proposed to take advantage of the advance in neural network based recommendation and made user and item embedding layer shared across multiple types of behaviors. Chen *et al.* [3] proposed a transfer-based prediction model for multiple behaviors with shared embedding matrices.

The second category of works [7, 22] regarded auxiliary user-item interactions as weaker signals of user preferences and proposed negative sampling based solutions for the multi-behaviour recommendation. Loni *et al.* [22] proposed to divide other types of user-item interaction into several levels, and proposed a negative sampler, which samples items from different levels with level-aware probability, to help train a matrix factorization based model. This is further extended by [7] to better exploit view data with an improved negative sampler.

In this work, we approach the task of multi-behavior recommendation without any prior knowledge on preference strengths, and propose an effective graph convolutional network based model to automatically learn the strength of each behaviour.

5.2 Graph-based Recommendation

Graph-based recommendation constructs input data in the form of graphs and designs graph-learning models for the recommendation. To be specific, the key goal of recommendation, predicting the probability of user-item interaction, can be regarded as a link prediction task on the graph, of which users and items are represented as nodes. Thus, the development of graph-based recommendation is in line with the progress of graph-learning models.

Some works [1, 8, 12, 25] apply random-walk based methods to graph-based recommendation. Baluja *et al.* [1] proposed to start a random walk from videos in a user's historical records through the videos' co-view graph. Then the reaching probability of video candidates is used for the prediction value of the target user's preference. Gori et al. [12] proposed an ItemRank model based on the PageRank algorithm [24] on item-item correlation graph for recommendation. Recently, the random-walk based recommendation is demonstrated to be very efficient in industrial systems [8]. Some other works [4, 5, 32, 41] rely on the graph-embedding techniques to learn user/item embeddings for recommendation. Chen *et al.* [4] proposed to build a graph consisting of four kinds of nodes—user, album, track, and artists— for music recommendation. User preference and query words are both encoded with latent vectors by graph-embedding techniques. Yang *et al.* [41] proposed to first capture user-item high-order connectivity on the graph and then combined it with traditional matrix factorization for recommendation. Chen *et al.* [5] further considered the neighboring-nodes based user-item similarity for learning embeddings of nodes for recommendation.

Recently, graph convolutional networks (GCN) [19, 34, 40] achieve a quantum leap in graph learning tasks. The basic idea of GCN is performing embedding propagation to capture node feature and graph structure at the same time. With the strong power of learning representations, GCN is widely applied in recommender systems [2, 36, 37, 39, 42, 43]. Ying *et al.* [42] proposed to apply GCN to pin-board graph with neighboring sampling technique for the pin-based recommendation. Berg *et al.* [2] introduced embedding propagation to five kinds of user-item rating matrices for factorization. This was further extended by [37] to more general implicit user-item interaction data. Recently, He *et al.* [14] proposed to remove the non-linear activation function and feature transformation in embedding propagation layers to improve the performance of CF tasks further. Besides these works on user-item interaction data, GCN is also utilized in more complicated recommendation tasks. Wu *et al.* [39] built graph from session interaction data and adopted GCN to learn representations for session-based recommendation. Wang *et al.* [36] utilized graph attentional networks for knowledge-aware recommendation with a unified graph containing user-item

relations and knowledge-graph relations. Zheng *et al.* [43] considered items' category and price as nodes in the graph, built a graph consisting of four types of nodes, and then applied GCN for the price-aware recommendation.

In our work, we build a heterogeneous graph based on users' multi-feedback and rely on the strong ability of graph convolutional networks in learning representation of users' and items' embedding vectors for recommendation.

6 CONCLUSIONS AND FUTURE WORK

In this work, we study the problem of multi-behavior recommendation that considers multiple types of user-item interactions. To fully model the different preference strengths reflected by different behaviors and various behavioral semantics, we propose a graph-based solution that re-constructs the multiple user-item interaction matrices into the unified graph. We then propose an MBGCN model that takes advantage of graph convolutional network's ability in learning node representations from complex graph structure. Extensive experimental results on real-world datasets demonstrate the superiority of our MBGCN model. Further ablation studies verify the effectiveness of modeling preference strength and behavioral semantics, respectively. We also evaluate the performance of cold-start users, and results confirm the applicability of MBGCN in real-world applications.

For future work, we plan to conduct experiments on online systems with A/B testing to evaluate the recommendation performance of our proposed solution. We also plan to explore the fine-grained multiple interactions in session level, which is also known as multiple micro-behaviors.

ACKNOWLEDGMENTS

This work was supported in part by The National Key Research and Development Program of China under grant 2018YFB1800804, the National Natural Science Foundation of China under U1936217, 61971267, 61972223, 61941117, 61861136003, Beijing Natural Science Foundation under L182038, Beijing National Research Center for Information Science and Technology under 20031887521, and research fund of Tsinghua University - Tencent Joint Laboratory for Internet Innovation Technology. This work was also supported by the National Natural Science Foundation of China (U19A2079).

REFERENCES

[1] Shumeet Baluja, Rohan Seth, Dharshi Sivakumar, Yushi Jing, Jay Yagnik, Shankar Kumar, Deepak Ravichandran, and Mohamed Aly. 2008. Video suggestion and discovery for youtube: taking random walks through the view graph. In *WWW*.

[2] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).

[3] Chong Chen, Min Zhang, Yongfeng Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2020. Efficient Heterogeneous Collaborative Filtering without Negative Sampling for Recommendation. In *AAAI*.

[4] Chih-Ming Chen, Ming-Feng Tsai, Yu-Ching Lin, and Yi-Hsuan Yang. 2016. Query-based music recommendations via preference embedding. In *Recsys*. ACM, 79–82.

[5] Chih-Ming Chen, Chuan-Ju Wang, Ming-Feng Tsai, and Yi-Hsuan Yang. 2019. Collaborative Similarity Embedding for Recommender Systems. In *WWW*.

[6] Zhiyong Cheng, Ying Ding, Xiangnan He, Lei Zhu, Xuemeng Song, and Mohan S Kankanhalli. 2018. Aˆ 3NCF: An Adaptive Aspect Attention Model for Rating Prediction.. In *IJCAI*. 3748–3754.

[7] Jingtao Ding, Guanghui Yu, Xiangnan He, Yuhan Quan, Yong Li, Tat-Seng Chua, Depeng Jin, and Jiajie Yu. 2018. Improving Implicit Recommender Systems with View Data.. In *IJCAI*. 3343–3349.

[8] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. 2018. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In *WWW*.

[9] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *WWW*.

[10] Chen Gao, Xiangnan He, Danhua Gan, Xiangning Chen, Fuli Feng, Yong Li, Tat-Seng Chua, Lina Yao, Yang Song, and Depeng Jin. 2019. Learning to Recommend with Multiple Cascading Behaviors. *TKDE* (2019).

[11] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*. 249–256.

[12] Marco Gori, Augusto Pucci, V Roma, and I Siena. 2007. Itemrank: A random-walk based scoring algorithm for recommender engines.. In *IJCAI*, Vol. 7. 2766–2771.

[13] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*. 1024–1034.

[14] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*.

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

[16] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*. 549–558.

[17] Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: factored item similarity models for top-n recommender systems. In *KDD*. 659–667.

[18] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR* (2015).

[19] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *ICLR* (2017).

[20] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*. 426–434.

[21] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).

[22] Babak Loni, Roberto Pagano, Martha Larson, and Alan Hanjalic. 2016. Bayesian personalized ranking with multi-channel user feedback. In *RecSys*. 361–364.

[23] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *ICDM*. IEEE, 497–506.

[24] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The pagerank citation ranking: Bringing order to the web.* Technical Report.

[25] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*. 701–710.

[26] Huihuai Qiu, Yun Liu, Guibing Guo, Zhu Sun, Jie Zhang, and Hai Thanh Nguyen. 2018. BPRH: Bayesian personalized ranking for heterogeneous implicit feedback. *Information Sciences* 453 (2018), 80–98.

[27] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*.

[28] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to recommender systems handbook. In *Recommender systems handbook*. Springer, 1–35.

[29] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. 285–295.

[30] 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.

[31] Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *KDD*. ACM, 650–658.

[32] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*. 1067–1077.

[33] Liang Tang, Bo Long, Bee-Chung Chen, and Deepak Agarwal. 2016. An empirical study on recommendation with multiple types of feedback. In *KDD*. 283–292.

[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. *ICLR* (2018).

[35] Hao Wang, Huawei Shen, Wentao Ouyang, and Xueqi Cheng. 2018. Exploiting POI-Specific Geographical Influence for Point-of-Interest Recommendation.. In *IJCAI*. 3877–3883.

[36] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *KDD*.

[37] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *SIGIR*. 165–174.

[38] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying Graph Convolutional Networks. In *ICML*. PMLR.

[39] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *AAAI*.

[40] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *ICLR* (2018).

[41] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. 2018. HOP-rec: high-order proximity for implicit recommendation. In *Recsys*.

[42] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *KDD*. ACM, 974–983.

[43] Yu Zheng, Chen Gao, Xiangnan He, Yong Li, and Depeng Jin. 2020. Price-aware Recommendation with Graph Convolutional Networks. In *ICDE*.