Interest-aware Message-Passing GCN for Recommendation

Fan Liu[†], Zhiyong Cheng[§*], Lei Zhu[‡], Zan Gao[§], Liqiang Nie[†*]
[†]School of Computer Science and Technology, Shandong University
[§]Shandong Artificial Intelligence Institute, Qilu University of Technology (Shandong Academy of Sciences)
[‡]School of Information Science and Engineering, Shandong Normal University
{liufancs,jason.zy.cheng,nieliqiang}@gmail.com

ABSTRACT

Graph Convolution Networks (GCNs) manifest great potential in recommendation. This is attributed to their capability on learning good user and item embeddings by exploiting the collaborative signals from the high-order neighbors. Like other GCN models, the GCN based recommendation models also suffer from the notorious over-smoothing problem – when stacking more layers, node embeddings become more similar and eventually indistinguishable, resulted in performance degradation. The recently proposed LightGCN and LR-GCN alleviate this problem to some extent, however, we argue that they overlook an important factor for the over-smoothing problem in recommendation, that is, high-order neighboring users with no common interests of a user can be also involved in the user's embedding learning in the graph convolution operation. As a result, the multi-layer graph convolution will make users with dissimilar interests have similar embeddings. In this paper, we propose a novel Interest-aware Message-Passing GCN (IMP-GCN) recommendation model, which performs high-order graph convolution inside subgraphs. The subgraph consists of users with similar interests and their interacted items. To form the subgraphs, we design an unsupervised subgraph generation module, which can effectively identify users with common interests by exploiting both user feature and graph structure. To this end, our model can avoid propagating negative information from high-order neighbors into embedding learning. Experimental results on three large-scale benchmark datasets show that our model can gain performance improvement by stacking more layers and outperform the state-of-the-art GCN-based recommendation models significantly.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Recommendation, Graph Convolution Networks, Message-Passing Strategy, Interest-aware, Subgraph

ACM Reference Format:
Fan Liu[†], Zhiyong Cheng[§*], Lei Zhu[‡], Zan Gao[§], Liqiang Nie[†*]. 2021. Interest-aware Message-Passing GCN for Recommendation. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3442381.3449986

* Corresponding author: Zhiyong Cheng and Liqiang Nie.

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3449986

1 INTRODUCTION

Recommendation system has become one of the most important techniques for various online platforms. It can not only provide personalized information for an specific user from overwhelming information, but also increase the revenue for service providers. Among them, Collaborative filtering (CF) based models [1, 15, 20, 38, 41] have made substantial progress in learning user and item representations by modeling historical user-item interactions. For example, matrix factorization (MF) can directly embed user/item as a feature vector and model the user-item interactions with inner product [1]. Neural collaborative filtering models replace the MF interaction function of inner product with nonlinear neural networks to learn better user and item representations [15].

Recently, GCN-based models [14, 21, 29, 32, 33] have achieved great success in recommendation due to the powerful capability on representation learning from non-Euclidean structure. The core of GCN-based models is to iteratively aggregate feature information from local graph neighbors. It has been proved to be an efficient way to distill additional information from graph structure, and thus improves user and item representation learning and alleviates the sparse problem. For example, NGCF [33] has proved that exploiting high-order connectivity can help alleviate the sparsity problem in recommendation. However, it is also well-recognized that GCNs suffer from the *over-smoothing* problem [33], because the graph convolution operation is actually a special kind of graph Laplacian smoothing [33], making node representations become indistinguishable after multi-layer graph convolution [40]. As a result, most current GCN based models obtain their peak performance by stacking only few layers (e.g., 2 or 3 layers), and continuing increasing the depth will lead to sharp performance degradation. In the domain of recommendation, Chen et al. [3] have empirically demonstrated that the user/item embeddings become more similar when stacking more layers in NGCF due to the over-smoothing effect. In other words, the preferences of different users become homogeneous, resulted in performance degradation in recommendation. Based on the observations, they proposed a LR-GCN model, which removes the non-linearities in GCNs to simply the network structure and introduced a residual network structure to alleviate the over-smoothing problem, achieving substantially improvement over NGCF on recommendation accuracy.

It is worth mentioning that the LightGCN proposed by He et al. [14] has a similar formulation as LR-GCN. With careful experimental studies, He et al. pointed out that the feature transformation and nonlinear activation have no positive effect (or even negative

effect due to the increase of training difficult) to the final performance. Therefore, they only keep the neighborhood aggregation in the LightGCN for collaborative filtering. Comparing to LR-GCN, LightGCN further removes the "self-loop" in the aggregation operation. Although LightGCN is not dedicatedly designed for tacking the over-smoothing problem, it has almost the same formulation as LR-GCN and thus can also alleviate the over-smoothing problem to some extent. In fact, both LR-GCN and LightGCN are consistent with the recent theories in simplifying GCNs [37] and can obtain the best performance with a deeper structure (e.g., 4 layers). Despite the two success GCN based models are designed for recommendation, we argue that they still design the model from the perspective of graph convolution, while have not well considered the over-smoothing problem in the domain of recommendation.

The GCN based recommendation model is built upon a user-item graph, in which the user and item are linked according to the historical user-item interactions. The user embedding is learned by iteratively aggregating messages passed from the neighboring (both user and item) nodes. Note that the passed messages are distilled from the embeddings of neighboring nodes. When stacking k layers, the information from the k-order neighbors, which are indirectly connected via items and users, are also involved in the embedding learning of a target node. An underlying assumption is that the collaborative signals from high-order neighbors are beneficial to the embedding learning. However, not all the information from high-order neighbors are positive in reality. In the user-item interaction graph, the high-order neighboring users could have no common or even contradictory interest with a target user. This is highly possible, especially when the graph is constructed based on implicit feedbacks (e.g., click). In fact, the implicit feedback is more widely used over the explicit feedbacks in modern recommendation systems. The core idea behind collaborative filtering is that similar users like similar items. Therefore, the collaborative signals that we would like to exploit should be from similar users (i.e., users with similar interests). However, existing GCN-based recommendation models have not distinguished the high-order neighbors, and just simply aggregate the messages from all those neighbors to update user embeddings. As a result, the embeddings of dissimilar users are also involved in the embedding learning of a target user, negatively affecting the performance. This is also a reason of the over-smoothing effect in the GCN-based recommendation models – making the embeddings of dissimilar users to be similar.

Motivated by the above considerations, in this paper, we propose a novel Interest-aware Messaging-Passing GCN (IMP-GCN) recommendation model, which groups users and their interacted items into different subgraphs and operates high-order graph convolutions inside subgraphs. More specific, we adopt the simplified network structure of LightGCN, as its effectiveness has been well demonstrated in [14] and it can alleviate the over-smoothing problem to some extent. The first-order graph convolution is the same as that of LightGCN. For the high-order graph convolution, only the messages from nodes in the same *subgraph* are exploited to learn the node embeddings. The subgraph is generated by a proposed graph generation module, which integrates users features and graph structure to identify users with similar interests, and then constructs the subgraphs by retaining those users and their interacted items. To this end, our model can filter out the negative information propagation in the high-order graph convolution operations for the embedding learning, and thus can keep the uniqueness of users by stacking more graph convolution layers. Extensive experiments have been conducted on three large-scale real-world datasets to validate the effectiveness of our model. Results show that our model outperforms the state-of-the-art methods by a large margin and can obtain better performance with more layers (till 7 layers) [1]. This indicates that our model can benefits from higher-order neighbors by excluding negative nodes. Besides, with deep analysis on the results, we found that the negative information in the embedding propagation is the major reason for the performance degradation of existing GCN-based recommendation models in deep structure. We released the codes and involved parameter settings to facilitate others to repeat this work [2].

In summary, the main contributes of this work are as follows:

- We step into the over-smoothing problem in existing GCN-based recommendation models and point out an overlooked factor: exploiting high-order neighbors indiscriminately makes the embeddings of users with dissimilar interests to be similar.
- We propose an IMP-GCN model which exploits high-order neighbors from the same subgraph, in which the user nodes share more similar interests than those in other subgraphs. It is proved to be effective on alleviating the over-smoothing problem.
- We design a subgraph generation module to group users and generate subgraphs from the user-item bipartite graph by considering users features and graph structure information.
- We conduct empirical studies on three benchmark datasets to evaluate the proposed IPM-GCN model. Results show that IPM-GCN can gain improvement by stacking more layers and learn better user/item embeddings, and thus outperforms the SOTA GCN-based recommendation models with a large margin.

2 METHODOLOGY

2.1 Recap

Let $\boldsymbol{A} \in \mathbb{R}^{N \times M}$ be the user-item interaction matrix, where N and M indicate the number of users and items, respectively. An nonzero entry $a_{ui} \in \boldsymbol{A}$ indicates that user $u \in \mathcal{U}$ has interacted with item $i \in \mathcal{I}$ before; otherwise, the entry is zero. A user-item bipartite graph $\mathcal{G} = (\mathcal{W}, \mathcal{E})$ can be constructed based on the interaction matrix, where the node set $\mathcal{W}$ consists of the two types of user nodes and item nodes and $\mathcal{E}$ represents for the set of edges. For a nonzero a_{ui}, there is an edge between the user u and item i. The above information is taken as the input of GCN model to learn the user and item representations by iteratively aggregating features from neighboring nodes in the bipartite graph.

Here we take LightGCN as an example to describe the GCN-based recommendation model, because it achieves the state-of-the-art performance with a very light design. Our model is also developed based on its design.[3] Let $\boldsymbol{e}_u^{(0)}$ denote the ID embedding of user u and $\boldsymbol{e}_i^{(0)}$ denote the ID embedding of item i, the graph convolution

[1] In experiments, we found that by stacking 7 layers, a user node almost reaches all the other users in three different datasets. Therefore, no more gain after stacking 7 layers.
[2] https://github.com/liufancs/IMP_GCN.
[3] Note that although LR-GCN was inspired by a different motivation, its final formulation is almost the same as LightGCN.

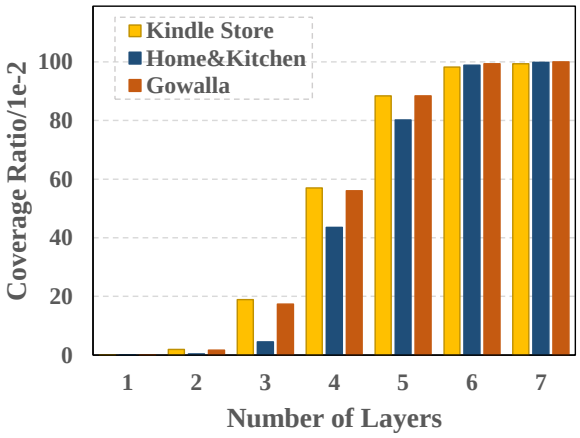


Figure 1: The average ratio of nodes involved in different layers of graph convolution on three datasets.

operation in LightGCN is described as follows:

$$\begin{aligned} \boldsymbol{e}_u^{(k)} &= \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \boldsymbol{e}_i^{(k-1)}, \\ \boldsymbol{e}_i^{(k)} &= \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \boldsymbol{e}_u^{(k-1)}, \end{aligned} \tag{1}$$

where $\boldsymbol{e}_u^{(k)}$ and $\boldsymbol{e}_i^{(k)}$ represent the embeddings of the user u and item i after k layers propagation, respectively; $\mathcal{N}_u$ denotes the set of items that interact with user u, and $\mathcal{N}_i$ denotes the set of users that interact with item i; $\frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}}$ is symmetric normalization terms, which can avoid the scale of embeddings increasing with graph convolution operations [18]. After K layers graph convolution, the final embeddings of a user u and an item i are the combination of their embeddings obtained at each layer in LightGCN:

$$\boldsymbol{e}_u = \sum_{k=0}^{K} \alpha_k \boldsymbol{e}_u^{(k)}; \boldsymbol{e}_i = \sum_{k=0}^{K} \alpha_k \boldsymbol{e}_i^{(k)}, \tag{2}$$

where $\alpha_k \geq 0$ is a hyper-parameter assigned to the k-th layer. It denotes the importance of this layer in constituting the final embedding. From Eq. 2, it is expected that after iteratively aggregating features from higher-order neighbors, the nodes will fail to preserve their own distinct features and their embeddings become more and more similar, leading to the over-smoothing problem. Besides, it does not distinguish the heterogeneous features of high-order nodes in the aggregation process. The noisy information from high-order neighbors could hurt the embedding learning. For example, the embeddings of users with no common interests or even contradictory interests in the high-order neighbors are aggregated to learn a target user's embedding via the graph convolution operation.

Fig. 1 shows the average coverage ratio of *the number of nodes that a target node reaches in the propagation by stacking different numbers of layers* to *all the nodes in the graph*. It can be seen that after 6- or 7-layer graph convolution, a node can almost receive information from all the other nodes in embedding propagation. Therefore, by aggregating information from all the connected high-order neighbors, it is unavoidable that the node embeddings become homogeneous in the current GCN-based models after stacking more layers, especially for the densely connected ones, whose embeddings will become more and more similar. In the recommendation scenario, this means the uniqueness of users will be neglected in deep structure.

Actually, current GCN-based recommendation models achieve their peak performance at most 3 or 4 layers [14, 37]. Besides the over-smoothing effect, we deem that a node also takes noisy or negative information in the embedding propagation process, which hurts the final performance. This is because a user's interests often span a range of items. Different users can have very different interests or even exhibit contradictory attitudes to some items. Without distinguishing those users, the embedding propagation may perform among users with very different interests to learn their embeddings in the graph convolution operation. To avoid the situation and alleviate the over-smoothing problem, it is important to group users with similar interests (and their interacted items) into subgraphs and constrain the embedding propagation to operate inside the subgraph. To achieve the goal we propose the interest-aware message-passing GCN model.

2.2 IMP-GCN MODEL

2.2.1 Interest-aware Message-passing Strategy. With constructing subgraphs, we would like that all the information propagated in a subgraph can contribute to the embedding learning of all the nodes in this subgraph. In other words, we aim to exclude the negative information propagation in the graph convolution operation using subgraphs. To achieve the goal, we rely on user nodes to form subgraphs in the user-item bipartite graph. The general idea is that users with more similar interests are grouped into a subgraph, and the items which directly linked to those users also belong to this subgraph. Therefore, each user only belongs to one subgraph, and an item can be associated with multiple subgraph. Let G_s with $s \in \{1, \cdots, N_s\}$ denotes a subgraph, where N_s is the number of subgraphs. In the next, we introduce the graph convolution operation in our model.

Because the direct interactions between users and items provide the most important and reliable information of user interests, in the first-order propagation, all the first-order neighbors are involved in the graph convolution operation. Let $\boldsymbol{e}_u^{(0)}$ and $\boldsymbol{e}_i^{(0)}$ denote the ID embeddings of user u and item i, respectively. The first-order graph convolution is:

$$\begin{aligned} \boldsymbol{e}_u^{(1)} &= \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \boldsymbol{e}_i^{(0)}, \\ \boldsymbol{e}_i^{(1)} &= \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \boldsymbol{e}_u^{(0)}, \end{aligned} \tag{3}$$

where $\boldsymbol{e}_u^{(1)}$ and $\boldsymbol{e}_i^{(1)}$ represent the first layer embeddings of the target user u and item i, respectively.

For the high-order graph convolution, to avoid introducing noisy information, a node in a subgraph can only exploit the information from its neighbor nodes in this subgraph. Because the items interacted by a user all belong to the subgraph of this user, the user can still receive information from all the linked items. However, for an item node, its direct user neighbors can be distributed in different

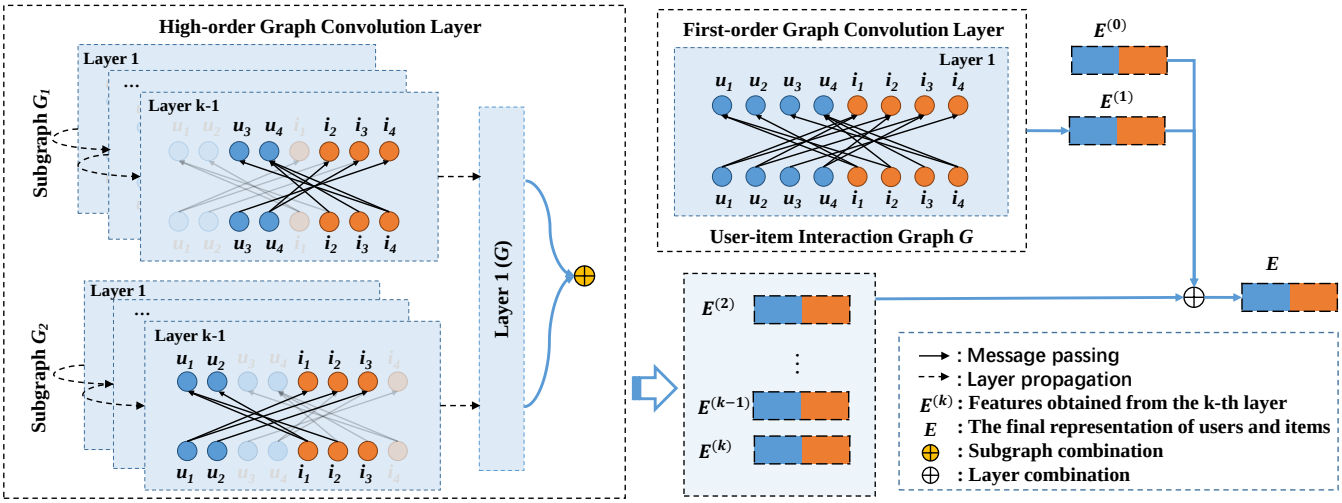


Figure 2: An overview of our IMP-GCN model with two subgraphs as illustration. In IMP-GCN, the first-order propagation operates on whole graph, and high-order propagation operates inside the subgraphs.

subgraphs. To learn the embeddings of an item i, for each subgraph G_s it belongs to, we learn an embedding for this item. Let $\boldsymbol{e}_{is}^{(k)}$ denotes the embedding of item i in subgraph s after k layers graph convolution, the high-order propagation in IMP-GCN is defined as:

$$\begin{aligned} \boldsymbol{e}_{u}^{(k+1)} &= \sum_{is \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} \boldsymbol{e}_{is}^{(k)}, \\ \boldsymbol{e}_{is}^{(k+1)} &= \sum_{u \in \mathcal{N}_i^s} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} \boldsymbol{e}_{u}^{(k)}. \end{aligned} \tag{4}$$

In this way, we guarantee that the embedding of a node learned in a subgraph only contributes to the embedding learning of other nodes in this subgraph. This can avoid the noisy information propagated from unrelated nodes. $\boldsymbol{e}_{is}^{(\cdot)}$ can be regarded as the features learned from the users with a similar interest in the subgraph G_s. This make senses since users with similar interests often prefer the same feature of an item. The final representation of an item i after k layers graph convolution is a combination of its embeddings learned in different subgraphs, i.e.,

$$\boldsymbol{e}_{i}^{(k)} = \sum_{s \in \mathcal{S}} \boldsymbol{e}_{is}^{(k)}, \tag{5}$$

where S is the subgraph set that item i belongs to.

2.2.2 Layer Combination and Prediction. We combine the embeddings obtained at each layer to form the final representation of user u and item i as Eq. 2. Similar to LightGCN, α_k is set uniformly as $1/(K+1)$ [14].

With the learned embeddings of users (i.e., $\boldsymbol{e}_u$) and items $\boldsymbol{e}_i$, given a user u and a target item i, the preference of the user to the item is computed by inner product:

$$\hat{r}_{uv} = \boldsymbol{e}_{u}^{T} \boldsymbol{e}_{i}. \tag{6}$$

Notice that other interaction functions can be also applied, such as Euclidean distance. Because the main focus of this work is to study the effects of distinguishing user interests in the graph convolution in the GCN-based recommendation model, we adopt the inner product as previous work [2, 33, 42] for fair comparisons in the empirical studies.

2.2.3 Matrix-form propagation rule. We implement our algorithm with the matrix form propagation rule (see [33] for more details), by which we can simultaneously update the representations of all users and items in a rather efficient way. It is a commonly used approach to make graph convolution network feasible for large-scale graph [26, 33]. Let $\boldsymbol{E}^{(0)}$ be the representations matrix for users ID and items ID; $\boldsymbol{E}^{(k)}$ represents the representation of users and items at the k-th layer. Similarly, $\boldsymbol{E}_s^{(k)}$ is defined as the representation of users and items at the k-th layer in subgraph G_s. As shown in Fig. 2, the first layer embedding propagation in our model can be described as follows:

$$\boldsymbol{E}^{(1)} = \mathcal{L}\boldsymbol{E}^{(0)}, \tag{7}$$

where $\mathcal{L}$ is the Laplacian matrix for the user-item interaction graph.

As we involve the subgraphs in high-order graph convolution layers,the embeddings propagation on subgraphs is formulated as follows:

$$\boldsymbol{E}_s^{(k-1)} = \mathcal{L}_s \boldsymbol{E}_s^{(k-2)}, \tag{8}$$

where $k \geqslant 2$; $\mathcal{L}_s$ represent the Laplacian matrix for the subgraph G_s. And then, the $(k-1)$-th layer embeddings are propagated on the user-item graph and obtained the embeddings in the k-th layer:

$$\boldsymbol{E}_s^{(k)} = \mathcal{L}\boldsymbol{E}_s^{(k-1)}. \tag{9}$$

We aggregate all the k-th layer embeddings involved different subgraphs to formulate the final k-th layer embeddings:

$$\boldsymbol{E}^{(k)} = \sum_{s \in G_s} \boldsymbol{E}_s^{(k)}. \tag{10}$$

Lastly, we combine all the layers' embeddings and get the final representations of users and items, this formulation keeps consistent with it in LightGCN [14]:

$$\boldsymbol{E} = \alpha_0 \boldsymbol{E}^{(0)} + \alpha_1 \boldsymbol{E}^{(1)} + \cdots + \alpha_K \boldsymbol{E}^{(K)} \tag{11}$$

2.2.4 Optimization. In this work, we target at the top-n recommendation, which aims to recommend a set of n top-ranked items matching the target user's preference. Compared to rating prediction, this is a more practical task in real commercial systems [27]. Similar to other rank-oriented recommendation works [33, 42], we adopt the pairwise learning method for optimization. To perform the pairwise learning, it needs to constructs a triplet of $\{u, i^+, i^-\}$, with *an observed interaction between u and* i^+ and *an unobserved interaction between u and* i^-. This method assumes that a positive item (i.e., i^+) should rank higher than an negative item (i.e., i^-). The objective function is formulated as:

$$\arg\min \sum_{(\mathbf{u},\mathbf{i^+},\mathbf{i^-})\in O} -\ln \phi(\hat{r}_{ui^+} - \hat{r}_{ui^-}) + \lambda \left\|\Theta\right\|_2^2 \tag{12}$$

where $O = \{(u, i^+, i^-)|(u, i^+) \in \mathcal{R}^+, (u, i^-) \in \mathcal{R}^-\}$ denotes the training set; $\mathcal{R}^+$ indicates the observed interactions between user u and i^+ in the training dataset, and $\mathcal{R}^-$ is the sampled unobserved interaction set. λ and Θ represent the regularization weight and the parameters of the model, respectively. The L_2 regularization is used to prevent overfitting.

The mini-batch Adam [17] is adopted to optimize the prediction model and update the model parameters. Specifically, for a batch of randomly sampled triples $(u, v^+, v^-) \in (O)$, the representation of those users and items are first learned by the propagation rules and then the model parameters are updated by using the gradients of the loss function.

2.3 Subgraph Generation Module

In this section, we introduce our proposed subgraph generation module which is designed to construct the subgraphs G_s with $s \in \{1, \cdots, N_s\}$ from a given input graph $\mathcal{G}$. Remind that the subgraphs are used to group users with common interests in our model. We formulate the user grouping as a classification task, i.e., each user is classified to a group. Specifically, each user is represented by a feature vector, which is a fusion of the graph structure and the ID embedding:

$$\boldsymbol{F_u} = \sigma(\boldsymbol{W_1}(\boldsymbol{e_u^{(0)}} + \boldsymbol{e_u^{(1)}}) + \boldsymbol{b_1}), \tag{13}$$

where $\boldsymbol{F_u}$ is the obtained user feature via feature fusion. $\boldsymbol{e_u^{(0)}}$ is the embedding of user ID and $\boldsymbol{e_u^{(1)}}$ is the feature obtained by aggregating local neighbor in the graph (i.e., the user embedding after the first layer propagation.). $\boldsymbol{W_1} \in R^{d\times d}$ and $\boldsymbol{b_1} \in R^{1\times d}$ are respectively the trainable weight matrix and bias vector of the fusion method. σ is the activation function. LeakyReLU [24] is adopted, because it can encode both positive and small negative signals. To classify the users into different subgraphs, we cast the obtained user feature to a prediction vector with a 2-layer neural networks:

$$\begin{aligned} \boldsymbol{U_h} &= \sigma(\boldsymbol{W_2}\boldsymbol{F_u} + \boldsymbol{b_2}), \\ \boldsymbol{U_o} &= \boldsymbol{W_3}\boldsymbol{U_h} + \boldsymbol{b_3}, \end{aligned} \tag{14}$$

where $\boldsymbol{U_o}$ is the prediction vector. The position of maximum value in U_o represents which group/subgraph the user belongs to. $\boldsymbol{W_2} \in R^{d\times d}$, $\boldsymbol{W_3} \in R^{d\times N_s}$ and $\boldsymbol{b_2} \in R^{1\times d}$, $\boldsymbol{b_3} \in R^{1\times N_s}$ are respectively the trainable weight matrices and bias vectors of the two layers. The dimension of the prediction vector dimensions is the same as the number of subgraphs, which is a pre-selected hyper-parameter.

Table 1: Basic statistics of the experimental datasets.

Dataset	#user	#item	#interactions	sparsity
Kindle Store	68,223	61,934	982,618	99.98%
Home&Kitchen	66,519	28,237	551,681	99.97%
Gowalla	29,858	40,981	1,027,370	99.92%

Note that it is an unsupervised method to classify users into different groups and thus does not need ground-truth label. For users with similar embeddings, Eq. 14 will generate similar prediction vector, namely, they will be classified into the same group. The subgraph generation aims to construct a matrix, which represents the user-item adjacency relation in a subgraph based on the user grouping results and the Laplacian matrix of the original user-item graph. For the matrix of each subgraph, according to the obtained user group information, we filter out the user-item adjacency relations in the Laplacian matrix of the original user-item graph if the corresponding users are not in the user group.

3 EXPERIMENTS

3.1 Experimental Setup

3.1.1 Data Description. To evaluate the effectiveness of IMP-GCN, we conducted experiments on three benchmark datasets: Amazon-Kindle Store, Amazon-Home&Kitchen and Gowalla. The first two datasets are from the public Amazon review dataset[4], which has been widely used for recommendation evaluation in previous studies. The third dataset is a check-in dataset collected from Gowalla, where users share their locations by checking-in. We followed the general setting in recommendation to filter users and items with few interactions. For all the datasets, we used the 10-core settings, i.e., retaining users and items with at least 10 interactions. The statistics of three datasets are shown in Table 1. As we can see, the datasets are of different sizes and sparsity levels, which are useful for analyzing the performance of our method and the competitors in different situations.

For each datasets, we randomly split it into training, validation, and testing set with the ratio 80:10:10 for each user. The observed user-item interactions were treated as positive instances. For the methods which adopt the pairwise learning strategy, we randomly sample a negative instance, that the user did not consume before, to pair with each positive instance.

3.1.2 Evaluation Metrics. For each user in the test set, we treat all the items that the user did not interact with as negative items. Two widely used evaluation metrics for top-n recommendation are adopted in our evaluation: Recall and Normalized Discounted Cumulative Gain [13]. For each metric, the performance is computed based on the top 20 results. Notice that the reported results are the average values across all the testing users.

3.1.3 Experimental Settings. We implemented our model with Tensorflow [5] and carefully tuned the key parameters. The embedding size is fixed to 64 for all models and the embedding parameters are initialized with the Xavier method [39]. We optimized our method

[4] http://jmcauley.ucsd.edu/data/amazon.
[5] https://www.tensorflow.org.

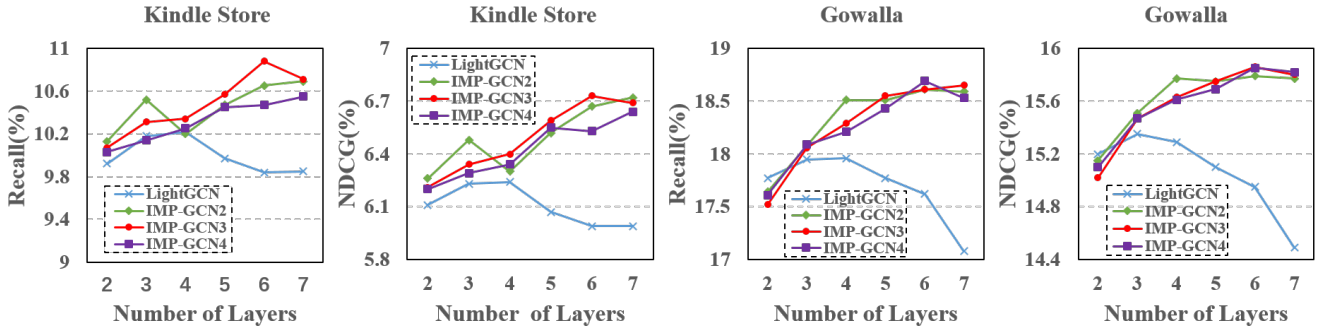


Figure 3: Results Comparison between IMP-GCN and LightGCN at different layers on Kindle Store and Gowalla. IMP-GCN$_2$, IMP-GCN$_3$, and IMP-GCN$_4$ represent IMP-GCN with 2, 3, and 4 subgraphs, respectively.

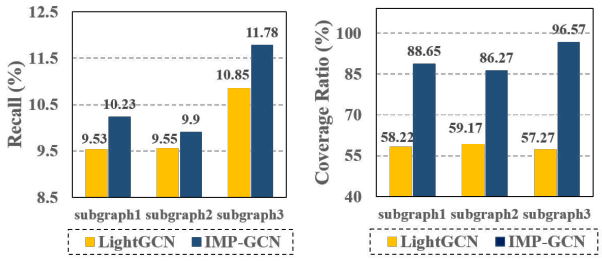
with Adam [17] and used the default learning rate of 0.001 and default mini-batch size of 1024 (on gowalla, we increased the mini-batch size to 2048 for speed). The L_2 regularization coefficient λ is searched in the range of $\{1e^{-6}, 1e^{-5}, \cdots, 1e^{-2}\}$. The early stopping and validation strategies are kept the same as those in LightGCN.

3.2 Study of IMP-GCN

In this section, we first evaluated the performance of our IPM-GCN model when stacking different layers in graph convolution. This is to examine whether our interest-aware message-passing strategy can alleviate the over-smoothing problem. In the next, we study the effects of the subgraph numbers on the performance of our model.

3.2.1 Effect of Layer Numbers. To investigate the effectiveness of IMP-GCN in deeper structure, we increased the model depth and performed detailed comparison with LightGCN. Since the adopted message-passing strategy is the same as LightGCN in the first-order convolution layer, we increased the layer number from 2 to 7. The experimental results are shown in Fig. 3, in which IMP-GCN$_2$, IMP-GCN$_3$ and IMP-GCN$_4$ indicate the model with 2, 3, and 4 subgraphs, respectively. We omitted the results on *Home&Kitchen* for space limitation, because they show exactly the same trend. From the results, we had some interesting observations.

Firstly, the proposed IMP-GCN outperforms LightGCN consistently when stacking more than 2 or 3 layers over both datasets. This indicates that our model can learn better embeddings by the interest-aware message-passing strategy. Secondly, the peak performance of LightGCN is obtained when stacking 3 or 4 layers, and increasing more layers will cause dramatic performance degradation, indicating it suffers from the over-smoothing problem in a deep structure. In contrast, IMP-GCN continues to achieve better performance with deeper structure (notice that when stacking more than 7 layers, a node already aggregates information from almost all the nodes, see Fig. 1. The results demonstrate the capability of our model on alleviating the over-smoothing problem. Moreover, it also 1) justifies our claim that exploiting information from all nodes indiscriminately causes the over-smoothing in GCN-based recommendation model, and 2) validates the effectiveness of our



(a) Recall on Kindle Store (b) Coverage Ratio on Kindle Store

Figure 4: Statistics of Recall and Coverage Ratio on Kindle Store in three subgraphs.

subgraph generation algorithm on classifying users with common interests.

3.2.2 Effect of Subgraph. The performance of IPM-GCN with different numbers (i.e., $\{2, 3, 4\}$) of subgraphs can also be observed in Fig. 3. From the results, we can see that the (1) IMP-GCN$_2$ with 2 subgraphs can obtain the best results when stacking no more than 3 layers. This is because a node in the subgraphs of IMP-GCN$_2$ can reach more nodes in short distance than the on in IMP-GCN$_3$ or IMP-GCN$_4$in the embedding propagation operation. (2) When stacking more than 3 layers, IMP-GCN$_3$ performs the best. After 3 layers graph convolution, the number of involved nodes increasing sharply in embedding propagation (see the examples in Fig. 1). On average, each node in IMP-GCN$_2$ should reach more nodes that the one in IMP-GCN$_3$ and IMP-GCN$_4$, however, the performance improvement of IMP-GCN$_2$ is smaller or even negative (on th Kindle Stores) than that of IMP-GCN$_3$ and IMP-GCN$_4$. This indicates that there is still noisy information in embedding propagation by discriminating user interests in a coarse-level (i.e., 2 subgraphs), negatively impacting the performance. Note that IMP-GCN$_3$ can still benefit from high-order neighbors. (3) With more subgraphs, on the one hand, IMP-GCN$_4$ can distinguish users with similar interests in a finer level and thus can better distill information from high-order neighbors; on the other hand, it also cuts more connections to other nodes, especially the ones in short distance

which provide more valuable information in embedding learning. As a result, when stacking more layers, its performance is only comparable to that of IMP-GCN$_2$. Therefore, there is a trade-off on selecting the number of subgraphs. We further studied the effects of subgraphs by analyzing the average coverage ratio of each node and the corresponding performance based on the LightGCN and our IPM-GCN model. Due to the space limitation, we only provide the results on Kindle Store and omit the performance $w.r.t$ ndcg which has the similar trend as recall. In this experiment, we used the LightGCN with 4 layers and IPM-GCN with 3 subgraphs[6] and 6 layers, which are their optimal setting on Kindle Store. The average recall and average cover ratio of each user in a subgraph based on LightGCN and IPM-GCN are shown in Fig. 4(a) and Fig. 4(b), respectively. Notably, by grouping users with similar interest in subgraphs to make information only propagate inside subgraphs, IPM-GCN can benefit from more layers of graph convolution and distill positive information from high-order neighors. In contrast, LightGCN is limited by the negative information from high-order neighbors and can only gain improvements over 4 layers. Comparing the performance of different subgraphs, we can see that with a higher coverage ratio, the performance of IPM-GCN increases clearly.

Another interesting finding is that, by stacking 6 layers, a user node in a subgraph almost connects to all the other nodes in the whole graph. This indicates that the users in a subgraph almost interact all the items in the graph (otherwise, the coverage ratio cannot be that high). More importantly, IPM-GCN can still achieve improvement with such high coverage without over-smoothing. This indicates that the embeddings of items learned in a graph contributes to the embedding learning of users in this graph, and the distilled information in a subgraph during graph convolution is useful for the embedding learning for all the nodes in this subgraph. It demonstrates the effectiveness of our interest-aware message-passing strategy and the subgraph generation algorithm.

3.3 Comparison with SOTA Methods

3.3.1 Baselines. To demonstrate the effectiveness, we compared our proposed method with several recently proposed competitive methods, including

- **NeuMF [15]**: It is a state-of-the-art neural collaborative filtering method. This method uses multiple hidden layers above the element-wise and concatenation of user and item embeddings to capture their non-linear feature interactions.
- **HOP-Rec [42]:** This method exploits the high-order user-item interactions by random walks to enrich the original training data. In experiments, we used the codes released by the authors [7].
- **CSE [2]:** This recently proposed graph-based model also exploits the high-order proximity in the user-item bipartite graph. Different from HOP-Rec, this method explores the user-user and item-item relations by random walks to improve the performance. We used the codes released by the authors (the same link as HOP-Rec).

[6]Number of users in the three groups G_1, G_2, G_3 are 3, 971, 3, 584, 6, 801, respectively.
[7]https://github.com/cnclabs/smore.

Table 2: Performance of our model and the competitors over three datasets. Noticed that the values are reported by percentage with '%' omitted.

Datasets	Kindle Store		Home&Kitchen		Gowalla	
Metrics	Recall	NDCG	Recall	NDCG	Recall	NDCG
NeuMF	4.96	2.06	1.34	0.62	12.96	11.21
CSE	7.65	4.54	1.93	0.91	13.85	11.51
HOP-Rec	7.96	4.58	1.98	0.94	14.11	12.70
GCMC	7.93	4.55	1.42	0.64	14.03	11.68
NGCF	8.25	5.09	2.14	0.96	15.62	13.35
LightGCN	**10.22**	**6.24**	**3.03**	**1.39**	**17.96**	**15.29**
IMP-GCN	**10.88***	**6.73***	**3.22***	**1.49***	**18.69***	**15.85***
Improv.	6.46%	7.85%	6.27%	7.19%	4.07%	3.66%

The symbol * denotes that the improvement is significant with $p - value < 0.05$ based on a two-tailed paired t-test.

- **GCMC [29]:** This method applies the GCN techniques on user-item bipartite graph and employs one convolutional layer to exploit the direct connections between users and items.
- **NGCF [33]**: This method explicitly encodes the collaborative signal in the form of high-order connectivities by performing embedding propagation in the user-item bipartite graph.
- **LightGCN [14]**: It is an simplified version of NGCF by removing the feature transformation and nonlinear activation module. It makes GCN-based methods more concise and appropriate for recommendation and achieves the state-of-the-art performance.

For fair comparisons, all the methods are optimized by the same pairwise learning strategy. We put great efforts to tune these methods based on the validation dataset and reported their best performance.

3.3.2 Overall Comparison. Table 2 shows the performance comparison results. The best and second best results were highlighted in bold. From the results, we had following observations.

The performance of NeuMF is relatively poor as it not explicitly leverages the high-order connectivities between users and items, resulting in suboptimal performance. For the graph-based methods, CSE makes use of the implicit associates of user-user and item-item similarities via high-order neighborhood proximity by performing random walks on the user-item interaction graph. GCMC obtains better performance over CSE, demonstrating the advantages of GCN-based approaches, which can exploit graph structure information. However, it does not perform well on *Home&Kitchen* because the useful information in neighbors cannot be efficiently aggregated. Hop-Rec outperforms the above methods on the three datasets, because it samples user-item interactions from high-order neighbors to enrich the training data. NGCF achieves consistent much better performance over the above baselines. This is because it adopts the GCN techniques to explicitly and directly exploit the high-order connectivities in the embedding. In contrast, the GCMC method only utilizes the first-order neighbors for representation learning; HOP-Rec and CSE leverage the high-order neighbors to enrich the training data rather than using them in embedding function for direct representation learning. This demonstrates the powerful representation learning capability of GCN and the importance of utilizing high-order information directly in representation learning.

Table 3: Performance of our model and its variants over three datasets. Noticed that the values are reported by percentage with '%' omitted.

Datasets	Kindle Store		Home&Kitchen		Gowalla	
Metrics	Recall	NDCG	Recall	NDCG	Recall	NDCG
IMP-GCN$_s$	10.57	6.63	3.14	1.43	18.61	15.61
IMP-GCN$_f$	10.19	6.40	2.97	1.31	17.84	15.11
IMP-GCN	**10.88**	**6.73**	**3.22**	**1.49**	**18.69**	**15.85**

Similar to the results reported in [14], LightGCN achieves substantially improvement over NGCF by simplifying it with the removal of two common designs in GCN.

IMP-GCN outperforms all the baselines consistently over all the datasets. In particular, compared to the strongest baseline in terms of NDCG@20, IMP-GCN can reach a relative improvement over LightGCN by 7.85%, 7.19%, 3.66% on *KindleStore*, *Home&Kitchen* and *Gowalla*, respectively. The great improvement over LightGCN demonstrates the importance of distinguishing nodes in high-order neighbors in the graph convolution operation, as well as the effectiveness of our proposed interest-aware message-passing strategy.

3.4 Ablation Study

In this section, we examined the contribution of different components in our model to the final performance by comparing IMP-GCN with the following two variants:

- **IMP-GCN$_s$**: This variant removes the graph structure information from the subgraph generation module (i.e., removing $\boldsymbol{e}_u^{(1)}$ in Eq. 13).
- **IMP-GCN$_f$**: In this variant, the first-order propagation is also performed inside each subgraph (i.e., The equation for $\boldsymbol{e}_i^{(1)}$ in Eq. 3 is replaced with $\sum_{s\in\mathcal{S}} \boldsymbol{e}_{is}^{(1)}$).

The results of two variants and IPM-GCN were reported in Table 3, in which the best results are highlighted in bold. IMP-GCN outperforms IMP-GCN$_s$ over all the datasets, which indicates the effectiveness of employing graph structure information in subgraph generation module. It is expected that IMP-GCN$_s$ obtains much better performance over IMP-GCN$_f$, because the first-order neighbors (i.e., the interaction between users and items) contributes the direct information for user and embedding in the collaborative filtering process. The results also demonstrate the reasonable design of our IPM-GCN model.

4 RELATED WORK

As one of the most important information retrieval techniques, recommendation has made tremendous progress in past decades. Among various recommendation approaches, the model-based collaborative filtering (CF) [5, 6, 14–16, 19, 20, 27, 32, 33] achieves a great success and becomes the mainstream recommendation technique. CF learns user and item embeddings by reconstructing the user-item interaction matrix. Earlier research efforts mainly focus on the shallow models, such as BPR [27], CML [16], matrix factorization (MF) [19]. Their success motivates the development of various variants via leveraging additional information (e.g., review [25], image [12], knowledge graph [30–32]) to deal with different tasks (e.g., context-aware [22], session-based [23]). With the rise of deep learning, it has also been widely applied in recommendation and exhibits great potential by either enhancing the user/item embedding learning or introducing non-linearity into the interaction function, promoting another peak development of recommendation technique. Many DL-based recommendation models have been proposed, such as NeuMF [15], Wide&Deep [4], and achieved better performance over traditional models.

Another research line is graph-based recommendation, which can explicitly exploit high-order proximity between users and items. Early approaches infer indirect preference by random walks in the graph to provide recommendation [7, 10, 11]. The recently proposed approaches exploit the user-item bipartite graph to enrich the user-item interactions [42, 44] and explore other types of collaborative relations, such as user-user and item-item similar ties [2, 44]. For example, HOP-Rec [42] uses random sample positive user-item interactions to enrich the training data by using random walks. WalkRanker [44] and CSE [2] performs random walks to explore the high-order proximity in user-user and item-item relations. As those methods rely on random walks to sample new interactions for model training, their performance heavily depends on the quality of generated interactions by random walks. As a result, these methods need carefully selection and tuning effects.

In recent years, Graph Convolution Networks (GCNs) have attracted increasing attention in recommendation due to the powerful capability on representation learning from non-Euclidean structure [8, 9, 14, 21, 29, 32–36, 43, 45]. And then, many GCN-based recommendation models have been developed. For example, GC-MC [29] employs one convolution layer to exploit the direct connections between users and items; PinSage [43] combines random walks with multiple graph convolution layers on the item-item graph for Pinterest image recommendation; MEIRec [8] utilizes metapath-guided neighbors to exploit rich structure information for intent recommendation; NGCF [33] exploits high-order proximity by propagating embeddings on the user-item interaction graph; instead of implicitly capturing the high-order connectivity through the propagation embedding, SMOG-CF [45] is proposed to directly capture the high-order connectivity between neighboring nodes at any order. Multi-GCCF [28] explicitly incorporates the user-user and item-item graphs, which is built upon the user-item bipartite graph, in the embedding learning process. Inspired by the study of simplifying GCN [37], researchers also introspect the complex design in GCN-based recommendation models. He at al. [14] pointed out that the two common designs feature transformation and nonlinear activation have no positive effects on the final performance, and proposed LightGCN which substantially improves the performance over NDCG. Meanwhile, Chen et al. [3] also proposed to remove the nonlinearity in the network and introduced a residual network to alleviate the over-smoothing problem in existing GCN-based recommendation models. In this paper, we move a step further on this research line. We claim that the indiscriminatively exploiting the high-order neighboring nodes is also an important reason for the over-smoothing problem for GCN-based recommendation model. A typical example is that two users with contradictory interests can be also connected via a k-order path in the user-item interaction graph. To tackle the problem, we propose an interest-aware

message-passing strategy to make the embedding propagation only happened inside a subgraph with similar interests.

5 CONCLUSION

In this work, we argued that exploiting high-order node indiscriminately would introduce negative information into the embedding propagation in the GCN-based recommendation models, causing the performance degradation when stacking more layers. We presented a IMP-GCN model which learns user and item embeddings by performing high-order graph convolution inside subgraphs. The subgraphs are formed by a designed subgraph generation algorithm that groups users with similar interests and their interacted items into the same graph. In IMP-GCN, the embedding of a node learned in a subgraph only contributes to the embedding learning of other nodes in this subgraph. In this way, IMP-GCN can effectively avoid taking the noisy information into the embedding learning. Experiments on large-scale real-world datasets demonstrate that IMP-GCN can gain improvements by stacking more layers to exploit information from higher-order neighbors, and achieve the state-of-the-art performance. The advantages of IMP-GCN indicate the importance of distinguishing high-order neighbors on tackling the over-smoothing problem in GCN models. We believe the insights in this study can shed light on the further development of graph-based recommendation models.

6 ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China, No.:61902223, No.:U1936203; the Innovation Teams in Colleges and Universities in Jinan, No.:2018GXRC014; the Shandong Provincial Natural Science Foundation, No.:ZR2019JQ23; Young creative team in universities of Shandong Province, No.:2020KJN012.

REFERENCES

[1] Robert M. Bell and Yehuda Koren. 2007. Lessons from the Netflix prize challenge. In *SIGKDD Explorations*. ACM, 75–79.
[2] Chih-Ming Chen, Chuan-Ju Wang, Ming-Feng Tsai, and Yi-Hsuan Yang. 2019. Collaborative Similarity Embedding for Recommender Systems. In *WWW*. ACM, 2637–2643.
[3] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*. AAAI Press, 27–34.
[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.
[5] Zhiyong Cheng, Xiaojun Chang, Lei Zhu, Rose C Kanjirathinkal, and Mohan Kankanhalli. 2019. MMALFM: Explainable recommendation by leveraging reviews and images. *TOIS* 37, 2 (2019), 16.
[6] Zhiyong Cheng, Ying Ding, Lei Zhu, and Kankanhalli Mohan. 2018. Aspect-aware latent factor model: Rating prediction with ratings and reviews. In *Proceedings of the 27th International Conference on World Wide Web*. IW3C2, 639–648.
[7] Fabian Christoffel, Bibek Paudel, Chris Newell, and Abraham Bernstein. 2015. Blockbusters and Wallflowers: Accurate, Diverse, and Scalable Recommendations with Random Walks. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 163–170.
[8] Shaohua Fan, Junxiong Zhu, Xiaotian Han, Chuan Shi, Linmei Hu, Biyu Ma, and Yongliang Li. 2019. Metapath-Guided Heterogeneous Graph Neural Network for Intent Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2478–2486.
[9] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *Proceedings of the 28th International Conference on World Wide Web*. IW3C2, 417–426.
[10] François Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. 2007. Random-Walk Computation of Similarities between Nodes of a Graph with Application to Collaborative Recommendation. *IEEE Trans. Knowl. Data Eng.* 19, 3 (2007), 355–369.
[11] Marco Gori and Augusto Pucci. 2007. ItemRank: A Random-Walk Based Scoring Algorithm for Recommender Engines. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*. Morgan Kaufmann Publishers Inc., 2766–2771.
[12] Ruining He and Julian McAuley. 2016. VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, 144–150.
[13] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. TriRank: Reviewaware Explainable Recommendation by Modeling Aspects. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*. ACM, 1661–1670.
[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 *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 639–648.
[15] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. IW3C2, 173–182.
[16] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *Proceedings of the 26th International Conference on World Wide Web*. IW3C2, 193–201.
[17] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.
[18] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*. OpenReview.net.
[19] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. In *IEEE Computer*, Vol. 42. 42–49.
[20] Fan Liu, Zhiyong Cheng, Changchang Sun, Yinglong Wang, Liqiang Nie, and Mohan Kankanhalli. 2019. User Diverse Preference Modeling by Multimodal Attentive Metric Learning. In *Proceedings of the 27th ACM International Conference on Multimedia*. ACM, 1526–1534.
[21] Fan Liu, Zhiyong Cheng, Lei Zhu, Chenghao Liu, and Liqiang Nie. 2020. An Attribute-aware Attentive GCN Model for Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2020), 1–12.
[22] Xin Liu and Wei Wu. 2015. Learning Context-Aware Latent Representations for Context-Aware Collaborative Filtering. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, 887–890.
[23] Yuanxing Liu, Zhaochun Ren, Wei-Nan Zhang, Wanxiang Che, Ting Liu, and Dawei Yin. 2020. Keywords Generation Improves E-Commerce Session-Based Recommendation. In *Proceedings of The Web Conference 2020*. Association for Computing Machinery, 1604–1614.
[24] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.
[25] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*. ACM, 165–172.
[26] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. 2018. DeepInf: Social Influence Prediction with Deep Learning.. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2110–2119.
[27] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.
[28] Jianing Sun, Yingxue Zhang, Chen Ma, Mark Coates, Huifeng Guo, Ruiming Tang, and Xiuqiang He. 2019. Multi-graph convolution collaborative filtering. In *Proceedings of IEEE International Conference on Data Mining*. 1306 – 1311.
[29] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2018. Graph Convolutional Matrix Completion. In *ACM SIGKDD: Deep Learning Day*. ACM.
[30] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. RippleNet: Propagating User Preferences on the Knowledge Graph for Recommender Systems. In *CIKM*. ACM, 417–426.
[31] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2019. Exploring High-Order User Preference on the Knowledge Graph for Recommender Systems. *ACM Trans. Inf. Syst.* 37, 3 (2019), 32:1–32:26.
[32] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 950–958.
[33] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 165–174.
[34] Yinwei Wei, Zhiyong Cheng, Xuzheng Yu, Zhou Zhao, Lei Zhu, and Liqiang Nie. 2019. Personalized Hashtag Recommendation for Micro-videos. In *Proceedings of the 27th ACM International Conference on Multimedia*. ACM, 1446–1454.

[35] Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, and Tat-Seng Chua. 2020. Graph-Refined Convolutional Network for Multimedia Recommendation with Implicit Feedback. In *Proceedings of the 28th ACM International Conference on Multimedia*. ACM, 3541–3549.

[36] Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, Richang Hong, and Tat-Seng Chua. 2019. MMGCN: Multi-modal graph convolution network for personalized recommendation of micro-video. In *Proceedings of the 27th ACM International Conference on Multimedia*. ACM, 1437–1445.

[37] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying Graph Convolutional Networks. PMLR, 6861–6871.

[38] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*. ACM, 153–162.

[39] Glorot Xavier and Bengio Yoshua. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. JMLR, 249–256.

[40] Xin Xin, Alexandros Karatzoglou, I. Arapakis, and J. Jose. 2020. Graph Highway Networks. *ArXiv* abs/2004.04635 (2020).

[41] HongJian Xue, XinYu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep matrix factorization models for recommender systems. In *Proceedings of the 26h International Joint Conference on Artificial Intelligence*. AAAI Press, 3203–3209.

[42] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. 2018. HOP-rec: high-order proximity for implicit recommendation. In *RecSys*. ACM, 140–144.

[43] 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 *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 974–983.

[44] Lu Yu, Chuxu Zhang, Shichao Pei, Guolei Sun, and Xiangliang Zhang. 2018. WalkRanker: A Unified Pairwise Ranking Model With Multiple Relations for Item Recommendation. In *IJCAI*. AAAI Press, 2596–2603.

[45] Hengrui Zhang and Julian McAuley. 2020. Stacked Mixed-Order Graph Convolutional Networks for Collaborative Filtering. In *Proceedings of the 2020 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, 73–81.