Neural Graph Collaborative Filtering

Xiang Wang
National University of Singapore
xiangwang@u.nus.edu

Xiangnan He*
University of Science and Technology of China
xiangnanhe@gmail.com

Meng Wang
Hefei University of Technology
eric.mengwang@gmail.com

Fuli Feng
National University of Singapore
fulifeng93@gmail.com

Tat-Seng Chua
National University of Singapore
dcscts@nus.edu.sg

ABSTRACT

Learning vector representations (*aka.* embeddings) of users and items lies at the core of modern recommender systems. Ranging from early matrix factorization to recently emerged deep learning based methods, existing efforts typically obtain a user's (or an item's) embedding by mapping from pre-existing features that describe the user (or the item), such as ID and attributes. We argue that an inherent drawback of such methods is that, the **collaborative signal**, which is latent in user-item interactions, is not encoded in the embedding process. As such, the resultant embeddings may not be sufficient to capture the collaborative filtering effect.

In this work, we propose to integrate the user-item interactions — more specifically the bipartite graph structure — into the embedding process. We develop a new recommendation framework *Neural Graph Collaborative Filtering* (NGCF), which exploits the user-item graph structure by propagating embeddings on it. This leads to the expressive modeling of **high-order connectivity** in user-item graph, effectively injecting the collaborative signal into the embedding process in an explicit manner. We conduct extensive experiments on three public benchmarks, demonstrating significant improvements over several state-of-the-art models like HOP-Rec [42] and Collaborative Memory Network [5]. Further analysis verifies the importance of embedding propagation for learning better user and item representations, justifying the rationality and effectiveness of NGCF. We publish our project at https://github.com/xiangwang1223/neural_graph_collaborative_filtering.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Collaborative Filtering, Recommendation, High-order Connectivity, Embedding Propagation, Neural Recommender Model

*Xiangnan He is the corresponding author.

ACM Reference Format:
Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19).* ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

Personalized recommendation is ubiquitous, having been applied to many online services such as E-commerce [43] and advertising [20]. At its core is an estimation of how likely a user will adopt an item based on the historical interactions (*e.g.*, purchases and clicks), known as collaborative filtering (CF) [31]. The basic assumption of CF is that similar users would exhibit similar preference on items; to implement it, a common paradigm is to parameterize users and items for reconstructing historical interactions, and predict user preference based on the parameters [17, 30].

Generally speaking, there are two key components in modern CF models — 1) *embedding*, which transforms users and items to vectorized representations, and 2) *interaction modeling*, which reconstructs historical interactions based on the embeddings. For example, matrix factorization (MF) directly embeds user/item ID as an vector and models user-item interaction with inner product [24]; collaborative deep learning methods extend the MF embedding function by integrating the deep representations learned from rich side information of items [36, 44]; neural collaborative filtering models replace the MF interaction function of inner product with nonlinear neural networks [17]; and translation-based CF models instead use Euclidean distance metric as the interaction function [11, 32], among others.

Despite their effectiveness, we argue that these methods are not sufficient to yield optimal embeddings for CF. The key reason is that the embedding function lacks an explicit encoding of the crucial **collaborative signal**, which is latent in user-item interactions to reveal the behavioral similarity between users (or items). To be more specific, most existing methods build the embedding function with the descriptive features only (*e.g.*, ID and attributes), while forgoing the modeling of user-item interactions — the user-item interactions are only used to define the learning objective for training the model [11, 30]. As a result, when the embeddings are insufficient in capturing CF, the methods have to rely on the interaction function to make up for the deficiency of suboptimal embeddings [17].

While it is intuitively useful to integrate user-item interactions into the embedding function, it is highly challenging to achieve this goal because the scale of interactions can easily reach

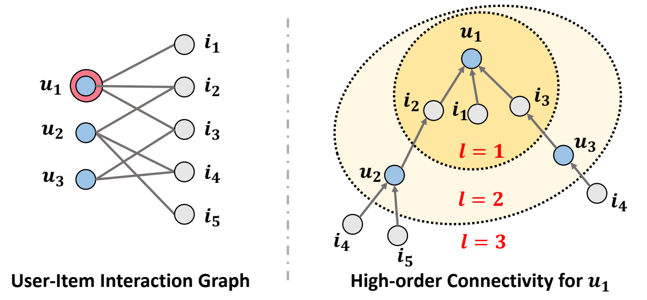


Figure 1: An illustration of the user-item interaction graph and the high-order connectivity. The node u_1 is the target user to provide recommendations for.

millions or even larger in real-world scenarios. This makes it prohibitive to efficiently extract useful collaborative signal to constitute an embedding in real-time. In this work, we tackle the challenge by exploiting the **high-order connectivity** from user-item interactions, a natural way that encodes collaborative signal in the bipartite interaction graph structure, to improve the embedding process.

Running Example. Figure 1 illustrates the concept of high-order connectivity. The user of interest for recommendation is u_1, labeled with the double circle in the left subfigure of user-item interaction graph. The right subfigure shows the tree structure that is expanded from u_1. The high-order connectivity denotes the path that reaches u_1 from any node with the path length l larger than 1. Such high-order connectivity contains rich semantics that carries collaborative signal. For example, the path $u_1 \leftarrow i_2 \leftarrow u_2$ indicates the behavior similarity between u_1 and u_2, as both users have interacted with i_2; the longer path $u_1 \leftarrow i_2 \leftarrow u_2 \leftarrow i_4$ suggests that u_1 is likely to adopt i_4, since her similar user u_2 has consumed i_4 before. Moreover, from the holistic view of $l = 3$, item i_4 is more likely to be of interest to u_1 than item i_5, since there are two paths connecting $<i_4, u_1>$, while only one path connects $<i_5, u_1>$.

Present Work. We propose to model the high-order connectivity information in the embedding function. Instead of expanding the interaction graph as a tree which has a high complexity, we design a neural network method to propagate embeddings recursively on the graph, as shown in Figure 2. It can be seen as constructing information flows in the embedding space. Specifically, we devise an **embedding propagation** layer, which updates a user's (or an item's) embedding by aggregating the embeddings of the interacted items (or users). By stacking multiple embedding propagation layers, we can enforce the embeddings to capture the collaborative signal in high-order connectivities. Taking Figure 1 as an example, stacking two layers captures the behavior similarity of $u_1 \leftarrow i_2 \leftarrow u_2$, stacking three layers captures the potential recommendations of $u_1 \leftarrow i_2 \leftarrow u_2 \leftarrow i_4$, and the strength of the information flow (which is estimated by the trainable weights between layers) determines the recommendation priority of i_4 and i_5. Since each layer produces an embedding for a user (or an item), we finally combine the embeddings of all layers to aggregate the collaborative signal learned from different orders of connectivities to form the embedding function. We conduct extensive experiments on three public benchmarks to verify the rationality and effectiveness of our *Neural Graph Collaborative Filtering* (NGCF) method.

Lastly, it is worth mentioning that although the high-order connectivity information has been considered in a very recent method named HOP-Rec [42], it is only exploited to enrich the training data. Specifically, the prediction model of HOP-Rec remains to be MF, while it is trained by optimizing a loss that is augmented with high-order connectivities. Distinct from HOP-Rec, we contribute a new technique to integrate high-order connectivities into the prediction model, which empirically yields better embeddings than HOP-Rec for CF.

To summarize, this work makes the following main contributions:

- We highlight the critical importance of explicitly exploiting the collaborative signal in the embedding function of model-based CF methods.
- We propose NGCF, a new recommendation framework based on graph neural network, which explicitly encodes the collaborative signal in the form of high-order connectivities by performing embedding propagation.
- We conduct empirical studies on three million-size real-world datasets. Extensive results demonstrate the state-of-the-art performance of NGCF and its effectiveness in improving the quality of embeddings with embedding propagation.

2 METHODOLOGY

We now present our NGCF as illustrated in Figure 2. There are three components in the framework: (1) the embedding layer to project user and item IDs to vector representations; (2) multiple embedding propagation layers that inject high-order connectivity modeling into the embeddings; and (3) the prediction layer which assembles multiple propagated representations and outputs the matching score of a user-item pair. We then discuss the time complexity of NGCF, and analyze the connections with existing methods.

2.1 Embedding Layer

Following embedding-based recommenders [17, 30], we associate the ID of user u and item i with the embeddings $\mathbf{e}_u \in \mathbb{R}^d$ and $\mathbf{e}_i \in \mathbb{R}^d$, where d is the embedding size. As a result, we establish the following embedding table:

$$\mathbf{E} = [\underbrace{\mathbf{e}_{u_1}, \cdots, \mathbf{e}_{u_N}}_{\text{users embeddings}}, \underbrace{\mathbf{e}_{i_1}, \cdots, \mathbf{e}_{i_M}}_{\text{item embeddings}}], \tag{1}$$

which can be viewed as the latent features of users and items to characterize their intrinsic properties.

2.2 Embedding Propagation Layers

Next we build upon the message-passing architecture of GNNs [4, 6, 7] in order to capture CF signal along the graph structure and enrich the representations of users and items.

First-order Propagation. Intuitively, the historical items provide direct evidence on a user's preference [21, 23]; meanwhile, the user group that consumes an item can be treated as the item's features, which reflects the collaborative similarity of two items. We build upon this basis to perform information propagation between the connected users and items.

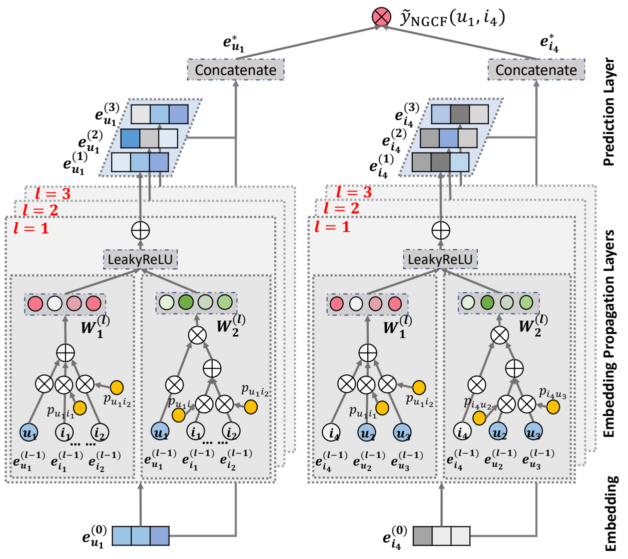


Figure 2: Schematic overview of NGCF, which updates the representations for user u_1 (left) and item i_4 (right). Wherein, the arrowed lines present the flow of information.

We formulate an embedding propagation layer, which consists of two phases: *message passing* and *message aggregation*. For a connected user-item pair (u, i), the message from i to u can be formulated as follows:

$$\mathbf{m}_{u\leftarrow i} = f(\mathbf{e}_i, \mathbf{e}_u, p_{ui}), \tag{2}$$

where $\mathbf{m}_{u\leftarrow i}$ is the message embedding (*i.e.,* the information being propagated), and $f(\cdot)$ is the information propagation function, which takes embeddings $\mathbf{e}_i$ and $\mathbf{e}_u$, and p_{ui} as input controls the decay factor on each propagation on edge (u, i).

In this work, we implement $f(\cdot)$ as:

$$\mathbf{m}_{u\leftarrow i} = \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}\Big(\mathbf{W}_1\mathbf{e}_i + \mathbf{W}_2(\mathbf{e}_i \odot \mathbf{e}_u)\Big), \tag{3}$$

where $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d'\times d}$ are the trainable weight matrices to distill useful information for propagation, and d' is the transformation size. Distinct from conventional graph convolution networks [4, 22, 33, 43] that consider the contribution of $\mathbf{e}_i$ only, here we additionally encode the interaction between $\mathbf{e}_i$ and $\mathbf{e}_u$ into the message being passed via $\mathbf{e}_i \odot \mathbf{e}_u$, where $\odot$ denotes the element-wise product. This makes the message dependent on the affinity between $\mathbf{e}_i$ and $\mathbf{e}_u$, *e.g.,* passing more messages from the similar items. This not only increases the model representation ability, but also boosts the performance for recommendation (evidences in our experiments Section 4.4.2).

Following graph convolution network [22], we set p_{ui} as the graph Laplacian norm $1/\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}$, where $\mathcal{N}_u$ and $\mathcal{N}_i$ denote the first-hop neighbors of user u and item i. From the viewpoint of representation learning, p_{ui} reflects how much the historical item contributes the user preference. From the viewpoint of message passing, p_{ui} can be interpreted as a discount factor, considering the messages being propagated should decay with path lengths [7].

The second phase of aggregation aims to assemble the messages propagated from u's neighborhood. Formally, we express the

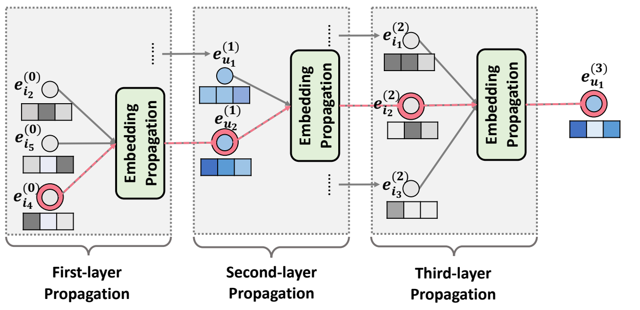


Figure 3: Illustration of third-order embedding propagation for user u_1. Best view in color.

aggregation function as,

$$\mathbf{e}_u^{(1)} = g\Big(\mathbf{e}_u, \{\mathbf{m}_{u\leftarrow i}|i \in \mathcal{N}_u\}\Big), \tag{4}$$

where $\mathbf{e}_u^{(1)}$ is the new representation of user u. In this work, we implement the aggregator by employing non-linear transformation on incoming messages:

$$\mathbf{e}_u^{(1)} = \sigma\Big(\mathbf{m}_{u\leftarrow u} + \sum_{i\in\mathcal{N}_u} \mathbf{m}_{u\leftarrow i}\Big), \tag{5}$$

where $\sigma(\cdot)$ is the activation function set as LeakyReLU [35] here. In addition to the messages passed from neighbors, we take the self-connection of u into consideration: $\mathbf{m}_{u\leftarrow u} = \mathbf{W}_1\mathbf{e}_u$, which retains the information of original features. Analogously, we can obtain the representation $\mathbf{e}_i^{(1)}$ for item i by propagating information from its connected users. To summarize, the advantage of the embedding propagation layer lies in explicitly exploiting the first-order connectivity information to relate user and item representations.

High-order Propagation. With the updated representations to describe users and items, we can further stack more propagation layers to explore the high-order connectivity information, and generate a single representation by assembling the messages from high-hop neighbors. Such high-order connectivities are crucial to encode the collaborative signal to estimate the relevance score between a user and item.

By stacking l embedding propagation layers, a user (and an item) is capable of receiving the messages propagated from its l-hop neighborhood. As Figure 2 displays, after l steps, the representation of user u is recursively formulated as:

$$\mathbf{e}_u^{(l)} = \sigma\Big(\mathbf{m}_{u\leftarrow u}^{(l)} + \sum_{i\in\mathcal{N}_u} \mathbf{m}_{u\leftarrow i}^{(l)}\Big), \tag{6}$$

wherein the message being propagated is defined as follows,

$$\mathbf{m}_{u\leftarrow i}^{(l)} = p_{ui}\Big(\mathbf{W}_1^{(l)}\mathbf{e}_i^{(l-1)} + \mathbf{W}_2^{(l)}\mathbf{e}_i^{(l-1)} \odot \mathbf{e}_u^{(l-1)}\Big), \tag{7}$$

where $\mathbf{W}_1^{(l)}, \mathbf{W}_2^{(l)}, \in \mathbb{R}^{d_l\times d_{l-1}}$ are the trainable transformation matrices, and d_l is the transformation size; $\mathbf{e}_i^{(l-1)}$ is the item representation generated from the previous message-passing steps, memorizing the messages from its (l-1)-hop neighbors. It further contributes to the representation of user u at layer l. Analogously, we can obtain the representation for item i at the layer l.

As Figure 3 shows, the collaborative signal like $u_1 \leftarrow i_2 \leftarrow u_2 \leftarrow i_4$ can be captured in the embedding propagation process. Furthermore, the message from i_4 is explicitly encoded in $\mathbf{e}_{u_1}^{(3)}$ (indicated by the red line). Clearly, the high-order embedding propagation seamlessly injects the collaborative signal into the representation learning process. We will analyze the complexity in Section 2.5.2.

2.3 Model Prediction

After propagating with L layers, we obtain multiple representations for user u, namely $\{\mathbf{e}_u^{(1)}, \cdots, \mathbf{e}_u^{(L)}\}$. Since the representations obtained in different layers emphasize the messages passed over different connections, they have different contributions in reflecting user preference. As such, we concatenate them to constitute the final embedding for a user; we do the same operation on items, concatenating the item representations $\{\mathbf{e}_i^{(1)}, \cdots, \mathbf{e}_i^{(L)}\}$ learned by different layers to get the final item embedding:

$$\mathbf{e}_u^* = \mathbf{e}_u^{(0)} \| \cdots \| \mathbf{e}_u^{(L)}, \quad \mathbf{e}_i^* = \mathbf{e}_i^{(0)} \| \cdots \| \mathbf{e}_i^{(L)}, \tag{8}$$

where $\|$ is the concatenation operation. By doing so, we not only enrich the initial embeddings with embedding propagation layers, but also allow controlling the range of propagation by adjusting L. Note that besides concatenation, other aggregators can also be applied, such as weighted average, max pooling, LSTM, *etc.*, which imply different assumptions in combining the connectivities of different order. The advantage of using concatenation lies in its simplicity, since it involves no additional parameters to learn, and it has been shown quite effectively in a recent work of graph neural networks [41], which refers to layer-aggregation mechanism.

Finally, we conduct the inner product to estimate the user's preference towards the target item:

$$\hat{y}_{\text{NGCF}}(u, i) = {\mathbf{e}_u^*}^\top \mathbf{e}_i^*. \tag{9}$$

In this work, we emphasize the embedding function learning thus only employ the simple interaction function of inner product. Other more complicated choices, such as neural network-based interaction functions [17], are left to explore in the future work.

2.4 Optimization

Similar to prior work [30], we opt for the BPR loss, which has been widely used to optimize recommendation models, *e.g.,* ACF [2], APR [16] *etc.* It is a pairwise loss that considers the relative order between observed and unobserved user-item interactions. Specifically, BPR assumes that the observed interactions, which are more reflective of a user's preferences, should be assigned higher prediction values than unobserved ones. The objective function for optimizing our model is as follows,

$$loss_{\text{NGCF}} = \sum_{(u,i,j)\in O} -\ln \mu(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda \|\Theta\|_2^2, \tag{10}$$

where $O = \{(u, i, j) | (u, i) \in \mathcal{R}^+, (u, j) \in \mathcal{R}^-\}$ denotes the training set, $\mathcal{R}^+$ indicates the observed (positive) interactions between user u and item j, while $\mathcal{R}^-$ is the sampled unobserved (negative) interaction set; $\mu(\cdot)$ is the sigmoid function; $\Theta = \{\mathbf{E}, \mathbf{W}_1^{(l)}, \mathbf{W}_2^{(l)}, \forall l \in \{1, \cdots, L\}\}$ is the model parameter set. Additionally, we conduct L_2 regularization parameterized by λ on Θ, in order to prevent overfitting. In terms of parameter size, the majority of our parameter cost comes from the user and item embeddings (*e.g.,* 4.5 million on Gowalla dataset), which is almost identical to that of MF; and the propagation layer weights are lightweight and negligible (*e.g.,* 24 thousand for three 64×64 layers on Gowalla dataset).

2.4.1 ***Training.*** We adopt mini-batch Adam to optimize the prediction model and update the model parameters. In particular, for a batch of randomly sampled triples $(u, i, j) \in O$, we establish their representations $[\mathbf{e}^{(0)}, \cdots, \mathbf{e}^{(L)}]$ after L steps of propagation, and then update model parameters by using the gradients of the loss function.

2.4.2 ***Message and Node Dropout.*** Although deep learning models have strong representation ability, they usually suffer from the overfitting. Dropout is an effective solution to prevent neural networks from overfitting. Following the prior work [33], we adopt two dropout techniques: *message dropout* and *node dropout*. Message dropout randomly drops out the outgoing messages. In particular, we propose to adopt dropout on the messages being propagated in Equation (7), with a probability p_1. As such, in the l-th propagation layer, only part of messages contributes to the new representations. We also conduct node dropout to randomly block a particular node and discard all its outgoing messages. Especially, for the l-th propagation layer, we randomly drop p_2 percent on the nodes of the Laplacian matrix, where p_2 is the dropout ratio.

While the message dropout endows the representations more robustness against the presence or absence of single connections between users and items, node dropout focuses on reducing the influences of particular users or items [33]. We perform experiments to investigate the role of message dropout and node dropout, analyzing their effectiveness in Section 4.4.3.

2.5 Discussions

In the subsection, we first show how NGCF generalizes SVD++ [23]. In what follows, we analyze the time complexity of NGCF.

2.5.1 ***NGCF Generalizes SVD++.*** SVD++ can be viewed as a special case of NGCF with no high-order propagation layer. In particular, we set L to one. Within the propagation layer, we disable the transformation matrix and nonlinear activation function. Thereafter, $\mathbf{e}_u^{(1)}$ and $\mathbf{e}_i^{(1)}$ are treated as the final representations for user u and item i, respectively. We term this simplified model as NGCF-SVD, which can be formulated as:

$$\hat{y}_{\text{NGCF-SVD}} = (\mathbf{e}_u + \sum_{i' \in \mathcal{N}_u} p_{ui'} \mathbf{e}_{i'})^\top (\mathbf{e}_i + \sum_{u' \in \mathcal{N}_i} p_{iu'} \mathbf{e}_i). \tag{11}$$

Clearly, by setting $p_{ui'}$ and $p_{u'i}$ as $1/\sqrt{|\mathcal{N}_u|}$ and 0 separately, we can exactly recover SVD++ model. Moreover, another widely-used item-based CF model, FISM [21], can be also seen as a special case of NGCF, wherein $p_{iu'}$ in Equation (11) is set as 0.

2.5.2 ***Complexity.*** As we can see, the layer-wise propagation rule is the main operation. For the l-th propagation layer, the matrix multiplication has computational complexity $O(|\mathcal{R}^+| d_l d_{l-1})$, where $|\mathcal{R}^+|$ denotes the number of nonzero entires in the Laplacian matrix; and d_l and d_{l-1} are the current and previous transformation size. For the prediction layer, only the inner product is involved, for which the time complexity of the whole training epoch is

$O(\sum_{l=1}^{L}|\mathcal{R}^{+}|d_l)$. Therefore, the overall complexity for evaluating NGCF is $O(\sum_{l=1}^{L}|\mathcal{R}^{+}|d_l d_{l-1} + \sum_{l=1}^{L}|\mathcal{R}^{+}|d_l)$. Empirically, under the same experimental settings (as explained in Section 4), MF and NGCF cost around $20s$ and $80s$ per epoch on Gowalla dataset for training, respectively; during inference, the time costs of MF and NGCF are $80s$ and $260s$ for all testing instances, respectively.

3 RELATED WORK

We review existing work on model-based CF, graph-based CF, and graph neural network-based methods, which are most relevant with this work. Here we highlight the differences with our NGCF.

3.1 Model-based CF Methods

Modern recommender systems [5, 17, 30, 44] parameterize users and items by vectorized representations and reconstruct user-item interaction data based on model parameters. For example, MF [24, 30] projects the ID of each user and item as an embedding vector, and conducts inner product between them to predict an interaction. To enhance the embedding function, much effort [13, 36, 44] has been devoted to incorporate side information like item content, social relations, and external knowledge graph. While inner product can force user and item embeddings of an observed interaction close to each other, its linearity makes it insufficient to reveal the complex and nonlinear relationships between users and items [17, 18]. Towards this end, recent efforts [11, 17, 18, 27, 40] focus on exploiting deep learning techniques to enhance the interaction function, so as to capture the nonlinear feature interactions between users and items. For instance, neural CF models, such as NeuMF [17], employ nonlinear neural networks as the interaction function; meanwhile, translation-based CF models, such LRML [32] and TransRec [11], instead model the interaction strength with Euclidean distance metrics.

Despite great success, we argue that the design of the embedding function is insufficient to yield optimal embeddings for CF, since the CF signals are only implicitly captured. Summarizing these methods, the embedding function transforms the descriptive features (*e.g.,* ID and attributes) to vectors, while the interaction function serves as a similarity measure on the vectors. Ideally, when user-item interactions are perfectly reconstructed, the transitivity property of behavior similarity could be captured. However, such transitivity effect showed in the Running Example is not explicitly encoded, thus there is no guarantee that the indirectly connected users and items are close in the embedding space. Without an explicit encoding of the CF signals, it is hard to obtain embeddings that meet the desired properties.

3.2 Graph-Based CF Methods

Another line of research [3, 15, 19, 29, 42] exploits the user-item interaction graph to infer user preference. Early efforts, such as ItemRank [9] and BiRank [15], adopt the idea of label propagation to capture the CF effect. To score items for a user, these methods define the labels as her interacted items, and propagate the labels on the graph. As the recommendation scores are obtained based on the structural reachness (which can be seen as a kind of similarity) between the historical items and the target item, these methods essentially belong to neighbor-based methods. However, these methods are conceptually inferior to model-based CF methods, since there lacks model parameters to optimize the objective function of recommendation.

The recently proposed method HOP-Rec [42] alleviates the problem by combining graph-based with embedding-based method. It first performs random walks to enrich the interactions of a user with multi-hop connected items. Then it trains MF with BPR objective based on the enriched user-item interaction data to build the recommender model. The superior performance of HOP-Rec over MF provides evidence that incorporating the connectivity information is beneficial to obtain better embeddings in capturing the CF effect. However, we argue that HOP-Rec does not fully explore the high-order connectivity, which is only utilized to enrich the training data[1], rather than directly contributing to the model's embedding function. Moreover, the performance of HOP-Rec depends heavily on the random walks, which require careful tuning efforts such as a proper setting of decay factor.

3.3 Graph Convolutional Networks

By devising a specialized graph convolution operation on user-item interaction graph (*cf.* Equation (3)), we make NGCF effective in exploiting the CF signal in high-order connectivities. Here we discuss existing recommendation methods that also employ graph convolution operations [33, 43, 45].

GC-MC [33] applies the graph convolution network (GCN) [22] on user-item graph, however it only employs one convolutional layer to exploit the direct connections between users and items. Hence it fails to reveal collaborative signal in high-order connectivities. PinSage [43] is an industrial solution that employs multiple graph convolution layers on item-item graph for Pinterest image recommendation. As such, the CF effect is captured on the level of item relations, rather than the collective user behaviors. SpectralCF [45] proposes a spectral convolution operation to discover all possible connectivity between users and items in the spectral domain. Through the eigen-decomposition of graph adjacency matrix, it can discover the connections between a user-item pair. However, the eigen-decomposition causes a high computational complexity, which is very time-consuming and difficult to support large-scale recommendation scenarios.

4 EXPERIMENTS

We perform experiments on three real-world datasets to evaluate our proposed method, especially the embedding propagation layer. We aim to answer the following research questions:

- **RQ1**: How does NGCF perform as compared with state-of-the-art CF methods?
- **RQ2**: How do different hyper-parameter settings (*e.g.,* depth of layer, embedding propagation layer, layer-aggregation mechanism, message dropout, and node dropout) affect NGCF?
- **RQ3**: How do the representations benefit from the high-order connectivity?

[1]The enriched trained data can be seen as a regularizer to the original training.

Table 1: Statistics of the datasets.

Dataset	#Users	#Items	#Interactions	Density
Gowalla	29, 858	40, 981	1, 027, 370	0.00084
Yelp2018	31, 831	40, 841	1, 666, 869	0.00128
Amazon-Book	52, 643	91, 599	2, 984, 108	0.00062

4.1 Dataset Description

To evaluate the effectiveness of NGCF, we conduct experiments on three benchmark datasets: Gowalla, Yelp2018, and Amazon-book, which are publicly accessible and vary in terms of domain, size, and sparsity. We summarize the statistics of three datasets in Table 1.

Gowalla[2]: This is the check-in dataset [26] obtained from Gowalla, where users share their locations by checking-in. To ensure the quality of the dataset, we use the 10-core setting [13], *i.e.,* retaining users and items with at least ten interactions.

Yelp2018[3]: This dataset is adopted from the 2018 edition of the Yelp challenge. Wherein, the local businesses like restaurants and bars are viewed as the items. We use the same 10-core setting in order to ensure data quality.

Amazon-book[4]: Amazon-review is a widely used dataset for product recommendation [12]. We select Amazon-book from the collection. Similarly, we use the 10-core setting to ensure that each user and item have at least ten interactions.

For each dataset, we randomly select 80% of historical interactions of each user to constitute the training set, and treat the remaining as the test set. From the training set, we randomly select 10% of interactions as validation set to tune hyper-parameters. For each observed user-item interaction, we treat it as a positive instance, and then conduct the negative sampling strategy to pair it with one negative item that the user did not consume before.

4.2 Experimental Settings

4.2.1 ***Evaluation Metrics.*** For each user in the test set, we treat all the items that the user has not interacted with as the negative items. Then each method outputs the user's preference scores over all the items, except the positive ones used in the training set. To evaluate the effectiveness of top-K recommendation and preference ranking, we adopt two widely-used evaluation protocols [17, 42]: recall@K and ndcg@K. By default, we set $K = 20$. We report the average metrics for all users in the test set.

4.2.2 ***Baselines.*** To demonstrate the effectiveness, we compare our proposed NGCF with the following methods:

- **MF** [30]: This is matrix factorization optimized by the Bayesian personalized ranking (BPR) loss, which exploits the user-item direct interactions only as the target value of interaction function.
- **NeuMF** [17]: The method is a state-of-the-art neural CF model which uses multiple hidden layers above the element-wise and concatenation of user and item embeddings to capture their non-linear feature interactions. Especially, we employ two-layered plain architecture, where the dimension of each hidden layer keeps the same.
- **CMN** [5]: It is a state-of-the-art memory-based model, where the user representation attentively combines the memory slots of neighboring users via the memory layers. Note that the first-order connections are used to find similar users who interacted with the same items.
- **HOP-Rec** [42]: This is a state-of-the-art graph-based model, where the high-order neighbors derived from random walks are exploited to enrich the user-item interaction data.
- **PinSage** [43]: PinSage is designed to employ GraphSAGE [10] on item-item graph. In this work, we apply it on user-item interaction graph. Especially, we employ two graph convolution layers as suggested in [43], and the hidden dimension is set equal to the embedding size.
- **GC-MC** [33]: This model adopts GCN [22] encoder to generate the representations for users and items, where only the first-order neighbors are considered. Hence one graph convolution layer, where the hidden dimension is set as the embedding size, is used as suggested in [33].

We also tried SpectralCF [45] but found that the eigen-decomposition leads to high time cost and resource cost, especially when the number of users and items is large. Hence, although it achieved promising performance in small datasets, we did not select it for comparison. For fair comparison, all methods optimize the BPR loss as shown in Equation (10).

4.2.3 ***Parameter Settings.*** We implement our NGCF model in Tensorflow and will release our code, data, and parameter settings upon acceptance. The embedding size is fixed to 64 for all models. For HOP-Rec, we search the steps of random walks in $\{1, 2, 3\}$ and tune the learning rate in $\{0.025, 0.020, 0.015, 0.010\}$. We optimize all models except HOP-Rec with the Adam optimizer, where the batch size is fixed at 1024. In terms of hyperparameters, we apply a grid search for hyperparameters: the learning rate is tuned amongst $\{0.0001, 0.0005, 0.001, 0.005\}$, the coefficient of L_2 normalization is searched in $\{10^{-5}, 10^{-4}, \cdots, 10^{1}, 10^{2}\}$, and the dropout ratio in $\{0.0, 0.1, \cdots, 0.8\}$. Besides, we employ the node dropout technique for GC-MC and NGCF, where the ratio is tuned in $\{0.0, 0.1, \cdots, 0.8\}$. We use the default Xavier initializer [8] to initialize the model parameters. Moreover, early stopping strategy is performed, *i.e.,* premature stopping if recall@20 on the test data does not increase for 50 successive epochs. To model the CF signal encoded in third-order connectivity, we set the depth of NGCF L as three. Without specification, we show the results of three embedding propagation layers, node dropout ratio of 0.0, and message dropout ratio of 0.1.

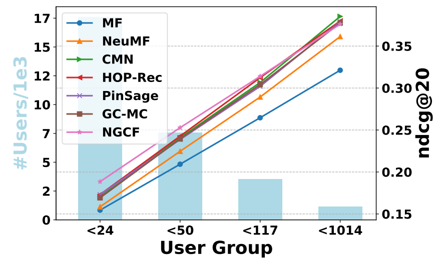
4.3 Performance Comparison (RQ1)

We start by comparing the performance of all the methods, and then explore how the modeling of high-order connectivity improves under the sparse settings.

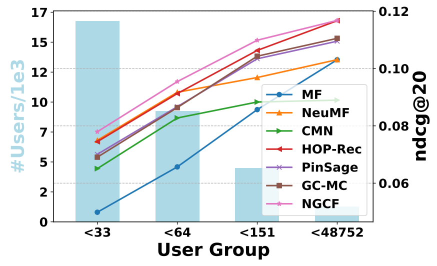
4.3.1 ***Overall Comparison.*** Table 2 reports the performance comparison results. We have the following observations:

- MF achieves poor performance on three datasets. This indicates that the inner product is insufficient to capture the complex relations between users and items, further limiting the performance. NeuMF consistently outperforms MF across all cases, demonstrating the importance of nonlinear feature

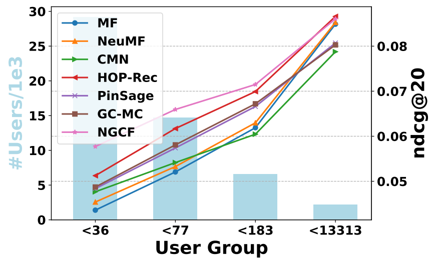
[2]https://snap.stanford.edu/data/loc-gowalla.html.
[3]https://www.yelp.com/dataset/challenge.
[4]http://jmcauley.ucsd.edu/data/amazon.



(a) ndcg on Gowalla



(b) ndcg on Yelp2018



(c) ndcg on Amazon-book

Figure 4: Performance comparison over the sparsity distribution of user groups on different datasets. Wherein, the background histograms indicate the number of users involved in each group, and the lines demonstrate the performance *w.r.t.* ndcg@20.

Table 2: Overall Performance Comparison.

	Gowalla		Yelp2018		Amazon-Book	
	recall	ndcg	recall	ndcg	recall	ndcg
MF	0.1291	0.1878	0.0317	0.0617	0.0250	0.0518
NeuMF	0.1326	0.1985	0.0331	0.0840	0.0253	0.0535
CMN	**0.1404**	**0.2129**	0.0364	0.0745	0.0267	0.0516
HOP-Rec	0.1399	0.2128	**0.0388**	**0.0857**	**0.0309**	**0.0606**
GC-MC	0.1395	0.1960	0.0365	0.0812	0.0288	0.0551
PinSage	0.1380	0.1947	0.0372	0.0803	0.0283	0.0545
NGCF	**0.1547***	**0.2237***	**0.0438***	**0.0926***	**0.0344***	**0.0630***
%Improv.	10.18%	5.07%	12.88%	8.05%	11.32%	3.96%
p-value	1.01e-4	5.38e-3	4.05e-3	2.00e-4	4.34e-2	7.26e-3

interactions between user and item embeddings. However, neither MF nor NeuMF explicitly models the connectivity in the embedding learning process, which could easily lead to suboptimal representations.

- Compared to MF and NeuMF, the performance of GC-MC verifies that incorporating the first-order neighbors can improve the representation learning. However, in Yelp2018, GC-MC underperforms NeuMF *w.r.t.* ndcg@20. The reason might be that GC-MC fails to fully explore the nonlinear feature interactions between users and items.
- CMN generally achieves better performance than GC-MC in most cases. Such improvement might be attributed to the neural attention mechanism, which can specify the attentive weight of each neighboring user, rather than the equal or heuristic weight used in GC-MC.
- PinSage slightly underperforms CMN in Gowalla and Amazon-Book, while performing much better in Yelp2018; meanwhile, HOP-Rec generally achieves remarkable improvements in most cases. It makes sense since PinSage introduces high-order connectivity in the embedding function, and HOP-Rec exploits high-order neighbors to enrich the training data, while CMN considers the similar users only. It therefore points to the positive effect of modeling the high-order connectivity or neighbors.
- NGCF consistently yields the best performance on all the datasets. In particular, NGCF improves over the strongest baselines *w.r.t.* recall@20 by 10.18%, 12.88%, and 11.32% in Gowalla, Yelp2018, and Amazon-Book, respectively. By stacking multiple embedding propagation layers, NGCF is capable of exploring the high-order connectivity in an explicit way, while CMN and GC-MC only utilize the first-order neighbors to guide the representation learning. This verifies the importance of capturing collaborative signal in the embedding function. Moreover, compared with PinSage, NGCF considers multi-grained representations to infer user preference, while PinSage only uses the output of the last layer. This demonstrates that different propagation layers encode different information in the representations. And the improvements over HOP-Rec indicate that explicit encoding CF in the embedding function can achieve better representations. We conduct one-sample t-tests and p-value < 0.05 indicates that the improvements of NGCF over the strongest baseline are statistically significant.

4.3.2 *Performance Comparison w.r.t. Interaction Sparsity Levels*.

The sparsity issue usually limits the expressiveness of recommender systems, since few interactions of inactive users are insufficient to generate high-quality representations. We investigate whether exploiting connectivity information helps to alleviate this issue.

Towards this end, we perform experiments over user groups of different sparsity levels. In particular, based on interaction number per user, we divide the test set into four groups, each of which has the same total interactions. Taking Gowalla dataset as an example, the interaction numbers per user are less than 24, 50, 117, and 1014 respectively. Figure 4 illustrates the results *w.r.t.* ndcg@20 on different user groups in Gowalla, Yelp2018, and Amazon-Book; we see a similar trend for performance *w.r.t.* recall@20 and omit the part due to the space limitation. We find that:

- NGCF and HOP-Rec consistently outperform all other baselines on all user groups. It demonstrates that exploiting high-order connectivity greatly facilitates the representation learning for inactive users, as the collaborative signal can be effectively captured. Hence, it might be promising to solve the sparsity issue in recommender systems, and we leave it in future work.
- Jointly analyzing Figures 4(a), 4(b), and 4(c), we observe that the improvements achieved in the first two groups (*e.g.,* 6.78% and 3.75% over the best baselines separately for < 24 and < 50 in Gowalla) are more significant than that of the other groups (*e.g.,* 0.49% for < 117 Gowalla groups). It verifies that the embedding propagation mechanism is beneficial to the relatively inactive users.

4.4 Study of NGCF (RQ2)

As the embedding propagation layer plays a pivotal role in NGCF, we investigate its impact on the performance. We start by exploring

Table 3: Effect of embedding propagation layer numbers (L).

	Gowalla		Yelp2018		Amazon-Book	
	recall	ndcg	recall	ndcg	recall	ndcg
NGCF-1	0.1511	0.2218	0.0417	0.0889	0.0315	0.0618
NGCF-2	0.1535	0.2238	0.0429	0.0909	0.0319	0.0622
NGCF-3	0.1547	0.2237	**0.0438**	**0.0926**	**0.0344**	0.0630
NGCF-4	**0.1560**	**0.2240**	0.0427	0.0907	0.0342	**0.0636**

the influence of layer numbers. We then study how the Laplacian matrix (*i.e.,* discounting factor p_{ui} between user u and item i) affects the performance. Moreover, we analyze the influences of key factors, such as node dropout and message dropout ratios. We also study the training process of NGCF.

4.4.1 ***Effect of Layer Numbers.***

To investigate whether NGCF can benefit from multiple embedding propagation layers, we vary the model depth. In particular, we search the layer numbers in the range of $\{1, 2, 3, 4\}$. Table 3 summarizes the experimental results, wherein NGCF-3 indicates the model with three embedding propagation layers, and similar notations for others. Jointly analyzing Tables 2 and 3, we have the following observations:

- Increasing the depth of NGCF substantially enhances the recommendation cases. Clearly, NGCF-2 and NGCF-3 achieve consistent improvement over NGCF-1 across all the board, which considers the first-order neighbors only. We attribute the improvement to the effective modeling of CF effect: collaborative user similarity and collaborative signal are carried by the second- and third-order connectivities, respectively.
- When further stacking propagation layer on the top of NGCF-3, we find that NGCF-4 leads to overfitting on Yelp2018 dataset. This might be caused by applying a too deep architecture might introduce noises to the representation learning. The marginal improvements on the other two datasets verifies that conducting three propagation layers is sufficient to capture the CF signal.
- When varying the number of propagation layers, NGCF is consistently superior to other methods across three datasets. It again verifies the effectiveness of NGCF, empirically showing that explicit modeling of high-order connectivity can greatly facilitate the recommendation task.

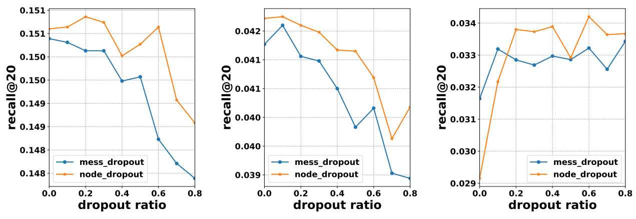
4.4.2 ***Effect of Embedding Propagation Layer and Layer-Aggregation Mechanism.***

To investigate how the embedding propagation (*i.e.,* graph convolution) layer affects the performance, we consider the variants of NGCF-1 that use different layers. In particular, we remove the representation interaction between a node and its neighbor from the message passing function (*cf.* Equation (3)) and set it as that of PinSage and GC-MC, termed NGCF-1_{PinSage} and NGCF-$1_{\text{GC-MC}}$ respectively. Moreover, following SVD++, we obtain one variant based on Equations (11), termed NGCF-$1_{\text{SVD++}}$. We show the experimental results in Table 4 and have the following findings:

- NGCF-1 is consistently superior to all variants. We attribute the improvements to the representation interactions (*i.e.,* $\mathbf{e}_u \odot \mathbf{e}_i$), which makes messages being propagated dependent on the affinity between $\mathbf{e}_i$ and $\mathbf{e}_u$ and functions like the attention mechanism [2, 34]. Whereas, all variants only take linear transformation into consideration. It hence verifies the rationality and effectiveness of our embedding propagation function.

Table 4: Effect of graph convolution layers.

	Gowalla		Yelp2018		Amazon-Book	
	recall	ndcg	recall	ndcg	recall	ndcg
NGCF-1	**0.1511**	**0.2218**	**0.0417**	**0.0889**	**0.0315**	**0.0618**
NGCF-$1_{\text{SVD++}}$	0.1447	0.2160	0.0380	0.0828	0.0277	0.0556
NGCF-$1_{\text{GC-MC}}$	0.1451	0.2165	0.0369	0.0812	0.0288	0.0562
NGCF-1_{PinSage}	0.1457	0.2170	0.0390	0.0845	0.0285	0.0563



(a) Gowalla (b) Yelp2018 (c) Amazon-Book

Figure 5: Effect of node dropout and message dropout ratios.

- In most cases, NGCF-$1_{\text{SVD++}}$ underperforms NGCF-1_{PinSage} and NGCF-$1_{\text{GC-MC}}$. It illustrates the importance of messages passed by the nodes themselves and the nonlinear transformation.
- Jointly analyzing Tables 2 and 4, we find that, when concatenating all layers' outputs together, NGCF-1_{PinSage} and NGCF-$1_{\text{GC-MC}}$ achieve better performance than PinSage and GC-MC, respectively. This emphasizes the significance of layer-aggregation mechanism, which is consistent with [41].

4.4.3 ***Effect of Dropout.***

Following the prior work [33], we employ node dropout and message dropout techniques to prevent NGCF from overfitting. Figure 5 plots the effect of message dropout ratio p_1 and node dropout ratio p_2 against different evaluation protocols on different datasets.

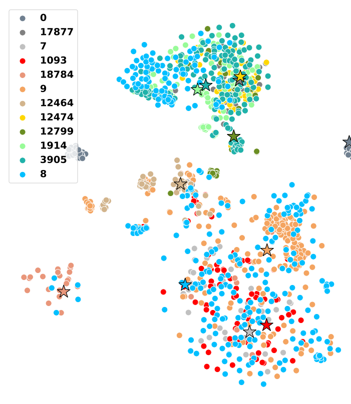
Between the two dropout strategies, node dropout offers better performance. Taking Gowalla as an example, setting p_2 as 0.2 leads to the highest recall@20 of 0.1514, which is better than that of message dropout 0.1506. One reason might be that dropping out all outgoing messages from particular users and items makes the representations robust against not only the influence of particular edges, but also the effect of nodes. Hence, node dropout is more effective than message dropout, which is consistent with the findings of prior effort [33]. We believe this is an interesting finding, which means that node dropout can be an effective strategy to address overfitting of graph neural networks.

4.4.4 ***Test Performance w.r.t. Epoch.***

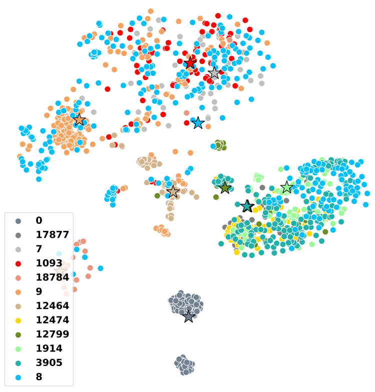
Figure 7 shows the test performance *w.r.t.* recall of each epoch of MF and NGCF. Due to the space limitation, we omit the performance *w.r.t.* ndcg which has the similar trend. We can see that, NGCF exhibits fast convergence than MF on three datasets. It is reasonable since indirectly connected users and items are involved when optimizing the interaction pairs in mini-batch. Such an observation demonstrates the better model capacity of NGCF and the effectiveness of performing embedding propagation in the embedding space.

4.5 Effect of High-order Connectivity (RQ3)

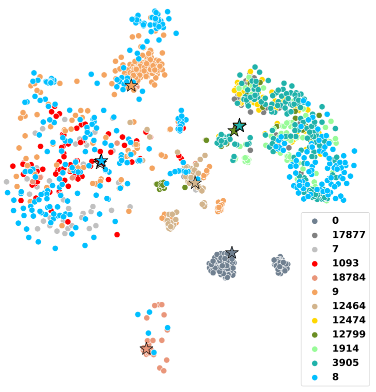
In this section, we attempt to understand how the embedding propagation layer facilitates the representation learning in the embedding space. Towards this end, for four user groups at different



(a) MF (NGCF-0)

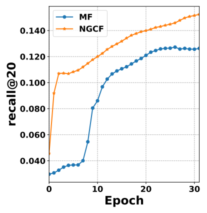


(b) NGCF-2

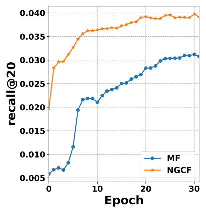


(c) NGCF-3

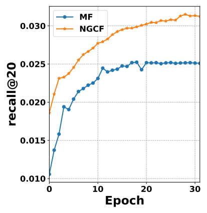
Figure 6: Visualization of the learned t-SNE transformed representations derived from MF-BPR, NGCF-2, and NGCF-3. Wherein, each star is a user, and the points with the same color denote the relevant items. Best view in color.



(a) Gowalla



(b) Yelp2018



(c) Amazon-Book

Figure 7: Test performance of each epoch of MF and NGCF.

sparsity levels as used in Section 4.3.2, we randomly selected three users per group, as well as that of their relevant items (from the test set, unseen in the training phase). We observe how these representations change when varying the depth of NGCF.

Figures 6(a), 6(b), and 6(c) show the visualization of the representations derived from MF (*i.e.,* NGCF-0), NGCF-2, and NGCF-3, respectively. Note that the items are from the test set, which are not paired with users in the training phase. There are two key observations:

- The connectivities of users and items are well reflected in the embedding space, that is, they are embedded into the near part of the space. In particular, the representations of NGCF-2 and NGCF-3 exhibit discernible clustering, meaning that the points with the same colors (*i.e.,* the items consumed by the same users) tend to form the clusters.
- Jointly analyzing the same users 17877, 18784, and 8 across Figures 6(a), 6(b), and 6(c), we find that, when stacking more embedding propagation layers, the embeddings of their historical items tend to be closer. It qualitatively verifies that the proposed embedding propagation layer is capable of injecting the collaborative user similarity (via NGCF-2) and explicit collaborative signal (via NGCF-3) into the representations.

5 CONCLUSION AND FUTURE WORK

In this work, we explicitly incorporated collaborative signal into the embedding function of model-based CF. We devised a new framework NGCF, which achieves the target by leveraging high-order connectivities in the user-item integration graph. The key of NGCF is the newly proposed embedding propagation layer, based on which we allow the embeddings of users and items interact with each other to harvest the collaborative signal. Extensive experiments on three real-world datasets demonstrate the rationality and effectiveness of injecting the user-item graph structure into the embedding learning process. In future, we will further improve NGCF by incorporating the attention mechanism [2, 34, 35] to learn variable weights for neighbors during embedding propagation and for the connectivities of different orders. This will be beneficial to model generalization and interpretability. Moreover, we are interested in exploring the adversarial learning [16] on user/item embedding and the graph structure for enhancing the robustness of NGCF.

This work represents an initial attempt to exploit structural knowledge with the message-passing mechanism in model-based CF and opens up new research possibilities. Specifically, there are many other forms of structural information can be useful for understanding user behaviors, such as the cross features in content-rich [14, 37] and context-aware [25, 28] recommendation, and item knowledge graph [1, 39, 44], and social networks [38]. For example, by integrating item knowledge graph with user-item graph, we can establish knowledge-aware connectivities between users and items, which help unveil user decision-making process in choosing items. We hope the development of NGCF is beneficial to the reasoning of user online behavior towards more effective and interpretable recommendation.

Acknowledgement: This research is part of NExT++ research and also supported by the Thousand Youth Talents Program 2018. NExT++ is supported supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@SG Funding Initiative.

REFERENCES

[1] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying Knowledge Graph Learning and Recommendation: Towards a Better Understanding of User Preferences. (2019).

[2] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention. In *SIGIR*. 335–344.

[3] Colin Cooper, Sang-Hyuk Lee, Tomasz Radzik, and Yiannis Siantos. 2014. Random walks in recommender systems: exact computation and simulations. In *WWW (Companion Volume)*. 811–816.

[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NeurIPS*. 3837–3845.

[5] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative Memory Network for Recommendation Systems. In *SIGIR*. 515–524.

[6] Alberto García-Durán and Mathias Niepert. 2017. Learning Graph Representations with Embedding Propagation. In *NeurIPS*. 5125–5136.

[7] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*. 1263–1272.

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

[9] Marco Gori and Augusto Pucci. 2007. ItemRank: A Random-Walk Based Scoring Algorithm for Recommender Engines. In *IJCAI*. 2766–2771.

[10] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NeurIPS*. 1025–1035.

[11] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based Recommendation. In *RecSys*. 161–169.

[12] Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *WWW*. 507–517.

[13] Ruining He and Julian McAuley. 2016. VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback. In *AAAI*. 144–150.

[14] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *SIGIR*. 355–364.

[15] Xiangnan He, Ming Gao, Min-Yen Kan, and Dingxian Wang. 2017. BiRank: Towards Ranking on Bipartite Graphs. *TKDE* 29, 1 (2017), 57–71.

[16] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial Personalized Ranking for Recommendation. In *SIGIR*. 355–364.

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

[18] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge J. Belongie, and Deborah Estrin. 2017. Collaborative Metric Learning. In *WWW*. 193–201.

[19] Mohsen Jamali and Martin Ester. 2009. *TrustWalker*: a random walk model for combining trust-based and item-based recommendation. In *KDD*. 397–406.

[20] Yu-Chin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware Factorization Machines for CTR Prediction. In *RecSys*. 43–50.

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

[22] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

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

[24] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42, 8 (2009), 30–37.

[25] Xiaopeng Li and James She. 2017. Collaborative Variational Autoencoder for Recommender Systems. In *KDD*. 305–314.

[26] Dawen Liang, Laurent Charlin, James McInerney, and David M. Blei. 2016. Modeling User Exposure in Recommendation. In *WWW*. 951–961.

[27] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *WWW*. 689–698.

[28] Jarana Manotumruksa, Craig Macdonald, and Iadh Ounis. 2018. A Contextual Attention Recurrent Architecture for Context-Aware Venue Recommendation. In *SIGIR*. 555–564.

[29] Athanasios N Nikolakopoulos and George Karypis. 2019. RecWalk: Nearly Uncoupled Random Walks for Top-N Recommendation. (2019).

[30] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. 452–461.

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

[32] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Latent relational metric learning via memory-based attention for collaborative ranking. In *WWW*. 729–739.

[33] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph Convolutional Matrix Completion. In *KDD*.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS*. 6000–6010.

[35] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[36] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *KDD*. 1235–1244.

[37] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2018. TEM: Tree-enhanced Embedding Model for Explainable Recommendation. In *WWW*. 1543–1552.

[38] Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2017. Item Silk Road: Recommending Items from Information Domains to Social Users. In *SIGIR*. 185–194.

[39] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. 2019. Explainable Reasoning over Knowledge Graphs for Recommendation. In *AAAI*.

[40] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In *WSDM*. 153–162.

[41] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML*, Vol. 80. 5449–5458.

[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*. 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 *KDD (Data Science track)*. 974–983.

[44] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *KDD*. 353–362.

[45] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S. Yu. 2018. Spectral collaborative filtering. In *RecSys*. 311–319.