Knowledge-Based Systems 228 (2021) 107251

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Dynamic evolution of multi-graph based collaborative filtering for recommendation systems

Hao Tang [a], Guoshuai Zhao [b,*], Xuxiao Bu [b], Xueming Qian [a,c]

[a] School of Information and Communication Engineering, Xi'an Jiaotong University, Xi'an 710049, China
[b] School of Software Engineering, Xi'an Jiaotong University, Xi'an 710049, China
[c] Ministry of Education Key Laboratory for Intelligent Networks and Network Security, Xi'an 710049, China

ARTICLE INFO

Article history:
Received 21 November 2020
Received in revised form 20 March 2021
Accepted 23 June 2021
Available online 25 June 2021

Keywords:
Multiple graphs
Collaborative filtering
Graph convolutional network
Rating prediction
Side information

ABSTRACT

The recommendation system is an important and widely used technology in the era of Big Data. Current methods have fused side information into it to alleviate the sparsity problem, one of the key problems of recommendation systems. However, not all the side information can be obtained with high quality, and the specific methods based on side information are not universal. In addition, side information has not been mined by the existing graph-based methods. To address these problems, we propose a Dynamic evolution of Multi-Graph Collaborative Filtering (DMGCF) model to mine and reuse side information. Specifically, we first construct user graph and item graph based on user-item bipartite graph and embeddings to exploit inter-user and inter-item relationships. The two new graphs simulate side information in latent space. Next, we perform a dual-path graph convolution network (GCN) on these three graphs for collaborative filtering. Then, a novel dynamic evolution mechanism is proposed to update and promote the embeddings and graphs collaboratively during the learning process, which produces better embeddings, user and item relationships, as well as the rating scores. We conduct a series of experiments on real-world datasets, and experimental results show the effectiveness of our approach.

1. Introduction

With the rapid development of the Internet, various information has been generated and "information overload" problem gradually emerges. Recommendation systems aiming to solve this problem have been developed vigorously and used widely in e-commerce, news, movies, music, travel, etc. Rating prediction is one of the key problems of the recommendation system. For instance, when listening to music, watching movies, shopping online, users always give them a rating to express their feelings. The website will also analyze users' preferences according to their ratings, and give them efficient and personalized recommendations. Recommendation systems serve as a bridge between information providers and users. Consequently, it is of great significance to pay attention to the study of this win-win technology.

Rating prediction has been widely considered and studied. Collaborative Filtering (CF) [1,2] is one of the most popular methods, but it faces the major defect of sparsity. Though the number of users or items is often tens of thousands, their corresponding rating pairs (user, item, rating) are relatively limited which results in a sparse rating matrix. To deal with this problem, side information [3] has been introduced to add more information of users and items. Side information can be formed by attributes and labels of users or items, such as the user's gender, age, income, geographical location, the movie's director, actors and release time. Besides, many types of side information which include users' reviews, social networks, contextual information, and knowledge graph [4–6] for items have been studied by many researchers.

Although side information is beneficial, it is difficult to obtain high-quality and a amount of side information. It also faces challenges to apply the information. For example, different networks need to be designed to fit various kinds of inputs. Review-based rating prediction methods need language or text processing foundation, while social network-based rating prediction methods require detailed analysis and design, such as social circles, trust. Rating prediction based on reviews, social networks, and locations have evolved into specialized recommendation methods, resulting in both the complexity of the algorithm and the limitation of its application. Therefore, the acquisition of information, the need for specialized algorithms, and application limitations become critical shortcomings of side information.

Recently, graph convolution network (GCN) [7–9] has become more and more widely used in recommendation systems. GCN is

* Corresponding author.
E-mail addresses: th1002@stu.xjtu.edu.cn (H. Tang), guoshuai.zhao@xjtu.edu.cn (G. Zhao), bo951024@stu.xjtu.edu.cn (X. Bu), qianxm@mail.xjtu.edu.cn (X. Qian).

https://doi.org/10.1016/j.knosys.2021.107251
0950-7051/

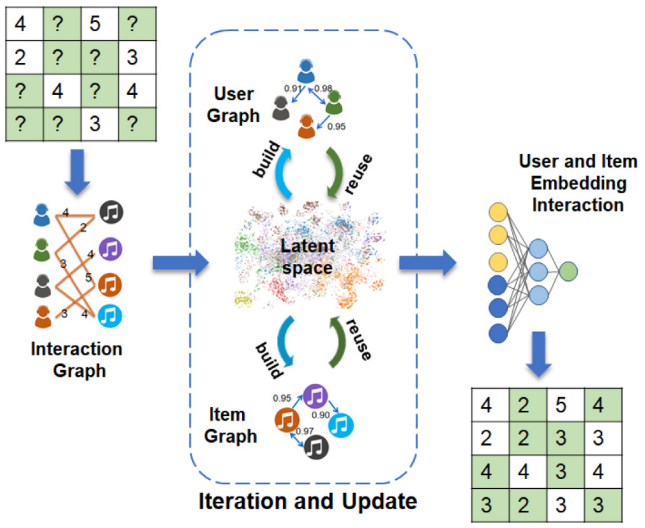


Fig. 1. The idea of our proposed model DMGCF. We propose multiple graphs that are self-generated and self-used, as well as the co-evolution of graphs and embeddings by the iteration and update process.

a kind of deep learning method for graphs, which has a strong ability of feature expression. The rating matrix can be viewed as a bipartite graph, and the rating prediction problem can be explored on the graph. There are three graphs in recommendation systems from the perspective of the graph: user-item interaction graph, social network on the user side, and knowledge graph on the item side. The latter two have rich connections and attributes, which can be used as supplementary information to improve the performance of the recommendation system. However, how to make use of these multiple graphs remains a challenge.

To make use of the powerful ability of GNN based on multiple graphs to improve the performance of the recommendation system and avoid the shortages of side information, we propose a new method in this paper. Instead of using real side information, we mine inter-user and inter-item relationships and build user graph and item graph to simulate and replace the side information in real-world. And a multi-graph based GCN method is formed. There are two main challenges. (1) It is a big difference that the user graph and item graph are homogeneous, directed, dynamic, and generated graphs while the user-item graph is not. We design a dual-path GCN to address the problem of different types of multiple graphs. (2) The right graphs are difficult to build just in once because the optimization is a gradual process. It is important that the generated graphs and embeddings could benefit from each other. How to achieve that in the optimization procedure is a vital problem. Thus, we propose a dynamic evolution mechanism. Under this mechanism, user and item graphs can be built many times to find the optimal results in the iterative learning process. Embeddings and the built graphs can be optimized together, and they can promote each other.

Therefore, a Dynamic evolution of Multi-Graph Collaborative Filtering (DMGCF) model is proposed which is shown in Fig. 1. Embeddings processed by GCN layers are used to generate the user graph and item graph. Next, the generated graphs are reused with the original graph forming multiple graphs based collaborative filtering. We build a dynamic evolution mechanism in which the embeddings and the generated graphs are both optimized in the training process. Furthermore, we gain a new type of dynamic graphs that are self-generated and evolved, while the traditional dynamic graphs are based on time series and only used as inputs.

DMGCF can be widely used in the real world for better performance because the required input is only the user-item graph. Many GCN backbones can be adopted and modified for DMGCF model and we use the Neural Graph Collaborative Filtering (NGCF) [10] framework. The main contributions of our approach are as follows:

- We propose a Dynamic evolution of Multi-Graph Collaborative Filtering (DMGCF) model. We generate multiple graphs with a dynamic evolution mechanism to simulate side information for better performance, especially when side information is unavailable.
- Our generated graphs are not only self-generated but also self-used. They are meaningful because they represent the inter-user and inter-item relationships. We design a dual-path GCN to address the problem of the different types of multiple graphs.
- We propose a dynamic evolution mechanism that makes the generated graphs and embeddings promote each other in the learning process. Better graphs help to get better embeddings, and vice versa. Experiments on real-world datasets verify the effectiveness of our model.

The rest of this paper is organized as follows: In Section 2, a series of related works are briefly reviewed and the differences between our work and them are addressed. In Section 3, the proposed DMGCF is introduced in detail. Experiments and discussions are given in Section 4 and conclusions are drawn in Section 5.

2. Related work

In this section, we introduce recent works related to our approach. Firstly, the collaborative filtering (CF) methods are introduced. The related methods using side information are discussed next. In addition, the graph convolution network (GCN) and its development in recommendation systems are introduced. Finally, the differences between DMGCF and previous works are analyzed.

2.1. Collaborative filtering

Collaborative filtering (CF) is the most commonly used method in the field of recommendation systems and rating prediction [1,2,11,12]. The idea of the CF algorithm is to recommend with the help of neighbors or similar users(items). So users or items can collaborate by helping each other. For example, recommend to the user what his/her friends like. CF methods can be divided into memory-based CF and model-based CF. Memory-based CF [1,2,12] recommendation algorithm obtains similar relationships between users or items according to the user-item rating matrix and finds the nearest neighbors for user or items. Predictions are made according to the neighbors. The recommendation accuracy depends on the adopted similarity measure, which is usually based on a suboptimal relation between users or between items. However, side information can be exploited for calculating or refining the similarities and thereby improving recommendations. Model-based CF [13–17] exploits data mining and machine learning algorithms by training specific models and gains high recommendation quality. Matrix factorization (MF) [13] is a typical method of model-based CF which models the rating matrix by two low-rank matrices. MF learns the potential relationship between users and items and predicts by inner product generally. Many variants of this model are proposed, such as Singular Value Decomposition (SVD) [14], Probabilistic Matrix Factorization (PMF) [15], Non-negative Matrix Factorization (NMF) [16], etc. Factorization Machines (FM) [18] which is a general predictor that can work with any real-valued feature vector can mimic many of the most successful factorization models including MF.

With the development of deep learning, more and more neural network methods for rating prediction are proposed [10,19–22]. Deep Matrix Factorization [19] constructs a user-item matrix with explicit ratings and non-preference implicit feedback, and then presents a matrix factorization model with neural network architecture to learn a common low dimensional space for the representations of users and items. AutoRec [20] proposes an autoencoder framework for collaborative filtering which is compact and has computational advantage. Neural collaborative filtering (NCF) [22] presents a neural network architecture to model latent features of users and items and devises a general framework for collaborative filtering based on neural networks. Recently, NGCF [10] introduces a graph-based neural collaborative filtering approach, but only the user-item graph is used. Thus, CF methods are still evolving, from traditional matrix decomposition to methods based on deep learning, and graph-based methods have been proposed recently. How to develop CF methods based on multiple graphs is a challenging problem we need to solve.

2.2. Side information based methods

More and more attention has been paid to the introduction of side information, or auxiliary information, to alleviate the sparsity problem and improve the performance of recommendation systems [23–26]. Side information contains a wide range of information related to users and items, such as age, gender, occupation of users and labels, attributes of items, etc. In recent years, locations, textual reviews, social networks, context information, knowledge graph have been widely explored in recommendation systems [27–31].

Location information is always used for kinds of recommendations [27,31–36]. Cheng et al. [33] propose a location-aware social music recommendation which considers users' location related contexts as well as global music popularity trends. VenueMusic [35] explores user's venue, which often includes surrounding atmosphere and related activities. Even complex correlations between clothing attributes and location attributes are mined to build location-oriented clothing recommendations for tourists [27]. Textual reviews are always used [37–39], for example, sentiment similarity from textual reviews is mined to build a sentiment-based rating prediction method [28]. And some of these review-based methods are able to provide interpretability by convolutional neural network and attention mechanism [37,38]. In addition, social information is widely used [40–43]. Hsu et al. [41] propose a framework to learn the degree of social correlation and rating prediction jointly. A dual graph attention network is proposed to model four interactions of social effects in user domain and item domain according to specific contexts [44]. Knowledge graph [4–6] provides rich relationships and attributes information for the items, and it has started to attract attention in the field of recommendation systems, but has not been used for rating prediction.

However, diversity of information means different inputs, and specific algorithms need to be designed, which make the recommendation system and the rating prediction task more complex and less adaptable. Besides, high-quality information is not always available and useful [45]. These shortcomings of side information inspire us to automatically generate and replace the side information in the real-world through relationship mining in latent space.

2.3. GCN based methods

Graph-based learning methods can make use of the interaction between users and items to make accurate recommendations. The complexity of graph data poses a major challenge to existing machine learning algorithms, so graph convolution network (GCN) becomes a new research hotspot [7–9]. GCN methods can be divided into two categories, spectral-based GCN and spatial-based GCN. The spectral-based method employs the theory of graph signal processing and introduces filters to define graph convolutions [46]. The spectrum-based GCN is limited in efficiency, generality, and flexibility while the space-based method is getting more and more attention [47,48]. The key idea of the space-based method is to use the information propagation mechanism on the graph to aggregate features from adjacent nodes, which is easier to understand and apply. The Message Passing Neural Network (MPNN) [47] is one of the representative methods which presents a simple but effective idea to update the state of each node after receiving the message from its neighbors.

GCN can be applied to almost every field of recommendation systems, such as rating prediction, top-n recommendation, session-based (sequential) recommendation, social recommendation, and so on [49–52]. SR-GNN [51] is proposed to obtain accurate item embeddings and take complex transitions of items into account with graph neural networks. Session sequences are modeled as graph-structured data so that GCN can capture complex transitions of items. GraphRec [52] is presented for social recommendations based on the GNN framework, which jointly captures interactions and opinions in the user-item graph and coherently models two graphs and heterogeneous strengths. Some graph-based methods for rating prediction are proposed recently [49,53,54]. STAR-GCN [49] proposes an architecture to learn node representations for boosting the performance in recommendation systems, especially in the cold start scenario. IGMC [53] achieves better performance and shows potential for transfer learning by mining local graph patterns. However, they are all based on one graph. sRMGCNN [54] proposes a multi-graph convolutional neural network architecture to learn meaningful statistical graph-structured patterns from users and items, but only the user graph and item graph are used which are always built by side information.

It is worth noting that there are bipartite graphs, attribute graphs, complex heterogeneous graphs, multi-source heterogeneous graphs and other types in the recommendation system [55,56]. How to extract useful information, reduce the noise, and integrate information from multi-source graphs or heterogeneous graphs is quite a challenge.

2.4. Differences with existing works

The relation and difference with the related works: DMGCF is a new development of the CF method and a new GCN method based on multiple graphs for rating prediction. It is a new approach by generating graphs to simulate side information in the recommendation system. Compared with similar works: Multiple graphs based on side information or the original rating matrix are used by sRMGCNN [54], while we adopt embedding-based and self-generated graphs without real side information in a new GCN model. The difference between NGCF [10] and our method is that we use ever-changing multiple graphs by mining relationships of users or item to gain a better CF prediction, while NGCF is based on only one graph. Besides, the dynamic evolution mechanism is proposed to make graphs and embeddings promote each other.

3. Methodology

In this section, the technical background and the overview of our method are introduced firstly. Then, the embedding layer, dual-path GCN based on multiple graphs and dynamic evolution mechanism are described in detail. The prediction layer and the loss function are introduced next. Finally, model complexities are analyzed.

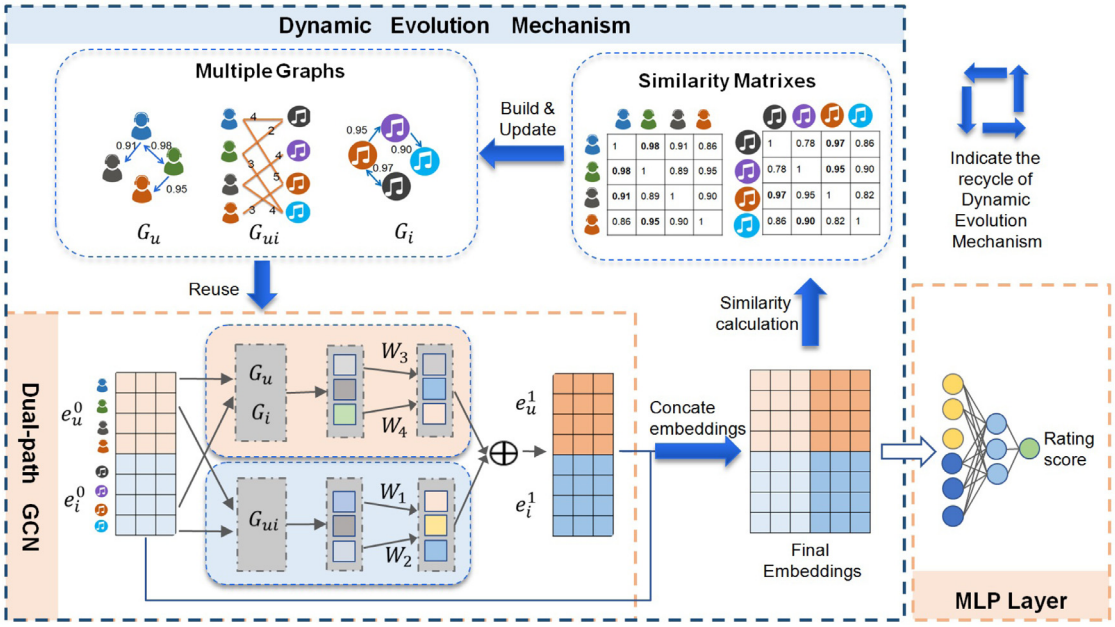


Fig. 2. An illustration of DMGCF model architecture. The dynamic evolution mechanism is the core of the method, which mainly includes the dual-path GCN layer, the process of multi-graph establishment and update. The MLP layer is used for feature interaction and prediction.

3.1. Preliminary

First, we briefly introduce some background methods, such as GCN [47–49] and NGCF [10], and an overview of our approach for the reader's convenience.

GCN: Briefly, the key idea of the space-based GCN method is to use the information propagation mechanism on the graph to aggregate features from neighbor nodes [47–49]. GCN can be described as:

$$h_i^{(k)} = \sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} h_j^{(k)} \tag{1}$$

$$h_i^{(k+1)} = \sigma(W h_i^{(k)}) \tag{2}$$

where $h_i^{(k)}$ denotes node features of node i in layer k; c_{ij} is a normalized constant, computed as $\sqrt{|\mathcal{N}_i|\,|\mathcal{N}_j|}$ (symmetric normalization) or $|\mathcal{N}_i|$ (left normalization) ; $\mathcal{N}$ denotes a set of neighbors of the node; W is learnable parameter matrices; σ is a non-linear function.

NGCF: NGCF [10] is a CF method with graph neural network, which exploits the user-item graph structure by propagating embeddings on it. NGCF explicitly encodes the collaborative signal by performing embedding propagation on more hops on the graph. First, each user and item is associated with an embedding for its representation. Let $e_u^{(0)}$, $e_u^{(0)}$ denote the initial embedding of user u and item i. Then NGCF propagates embeddings on the user-item interaction graph as:

$$e_u^{(k+1)} = \sigma\Big(\sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u \mathcal{N}_i|}} (W_1 e_i^{(k)} + W_2(e_i^{(k)} \odot e_u^{(k)})) + W_1 e_u^{(k)}\Big) \tag{3}$$

$$e_i^{(k+1)} = \sigma\Big(\sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u \mathcal{N}_i|}} (W_1 e_u^{(k)} + W_2(e_u^{(k)} \odot e_i^{(k)})) + W_1 e_i^{(k)}\Big) \tag{4}$$

where $e_i^{(k)}$, $e_i^{(k)}$ respectively denote the refined embedding of u and i after k layers propagation; W_1 and W_2 are trainable weight matrices to perform feature transformation in each layer. By stacking k embedding propagation layers, a user (or an item) is capable of receiving the messages propagated from its k-hop neighbors. It then concatenates these k+1 embeddings to obtain the final user embedding and item embedding, using the inner product to generate the prediction score of a user-item pair.

Overview: Our proposed multi-graph based model with a dynamic evolution mechanism is shown in the center of Fig. 2. We first create embeddings of users and items, e_u^0, e_i^0 in the embedding layer. We connect them to a dual-path GCN layer designed for multiple graphs, and new features e_u^1, e_i^1 can be gained through the first GCN layer (e_u^2, e_i^2 learned from the second GCN layer, etc. For brevity, only one GCN layer is illustrated in Fig. 2). After that, all the embeddings are concatenated as the final embeddings of users and items. The final representation is put together into a Multi-Layer Perceptron (MLP) and converted into rating scores. This is the general process of graph-based CF method.

Based on the final embeddings of users and items, the similarity matrices are calculated by the cosine similarity metric. The top-k similar users or items are selected in rows of the similarity matrices to build the user graph and item graph, G_u, G_i. The similarity values are recorded as the edge weights of the graph. Along with the original user-item graph, G_{ui}, multiple graphs are constructed and sent to the dual-path GCN embedding propagation layers for CF. In this way G_u and G_i are reused as collaborative signals to learn new embeddings. Thus, all the embeddings and the generated G_u, G_i can be updated and promoted in the dynamic evolution mechanism until the end of the training process.

3.2. Embedding layer

Following mainstream recommender models [10,22], we first create embeddings of users and items as their representation, $e_u, e_i \in \mathbb{R}^d$, where d is the embedding dimension. They are parameters to be learned and optimized through the training process. Different dimensions of embeddings indicate different features or preferences of the users or items in the recommendation system.

3.3. Dual-path GCN

As can be seen from the following analysis, G_{ui} is different from G_u and G_i while the latter two are similar. We design

a new branch for G_u and G_i, thus forming a dual-path GCN. Fig. 2 shows that there are two paths in the "Dual-path GCN" section, with the new path added for the generated graphs at the top. The message-passing architecture is applied by spatial-based GCN [10,48]. We describe the dual-path GCN in two steps as NGCF, namely, message construction and message aggregation.

3.3.1. Message construction

Three graphs, G_{ui}, G_u and G_i, are shown in the "Multiple Graphs" part in Fig. 2 which represent the relationship between users and items, the relationship among users, and the relationship among items respectively. It is necessary to point out that at the beginning of training G_u and G_i are initialized to zeros matrices. They are generated after some training epochs, but their formulas are consistent.

Users and items share the same message construction and information aggregation mechanisms and users are taken as example. For any user-item pair (u, i), the feature of the item i is passed as a message to u through a neural network with parameter W_1. Meanwhile, the affinity of the user and item is modeled by element-wise product method to pass more messages from similar items, with parameter W_2 used for feature learning. $m_{u\leftarrow i}$ represents messages from each i to u, which is the original path. Similarly, for the new graph G_u, we passed the information of the new neighbor and modeled the affinity between the two users by using W_3, W_4. $m_{u\leftarrow u'}$ represents messages from each neighbor u' to u, which is in the new path. All the neighbors of u in G_{ui} and G_u pass information according to the above mentioned process. Items have the same messages, $m_{i\leftarrow u}$, $m_{i\leftarrow i'}$, as the users. All message construction functions are expressed as:

$$m_{u\leftarrow i} = \frac{1}{\sqrt{|\mathcal{N}_u\mathcal{N}_i|}}(W_1 e_i + W_2(e_i \odot e_u)) \tag{5}$$

$$m_{u\leftarrow u'} = \frac{1}{|\mathcal{N}_{u'}|}(W_3 e_{u'} + W_4(e_{u'} \odot e_u)) \tag{6}$$

$$m_{i\leftarrow u} = \frac{1}{\sqrt{|\mathcal{N}_i\mathcal{N}_u|}}(W_1 e_u + W_2(e_u \odot e_i)) \tag{7}$$

$$m_{i\leftarrow i'} = \frac{1}{|\mathcal{N}_{i'}|}(W_3 e_{i'} + W_4(e_{i'} \odot e_i)) \tag{8}$$

where $\mathcal{N}$ is normalization constant, $\mathcal{N}_u, \mathcal{N}_i$ normalized by the neighbors of user u and item i in G_{ui}, $\mathcal{N}_{u'}, \mathcal{N}_{i'}$ normalized by the neighbors of user u in G_u and the neighbors of item i in G_i. And $\odot$ denotes the element-wise product.

We adopt a new path of GCN with parameters W_3, W_4 to learn the different patterns and pass collaborative signals in G_u and G_i, as shown in formulas (6) and (8). The new path is designed to solve the challenge of large differences between multiple graphs:

(1) Differences in graph types: G_u and G_i are directed and dynamic graphs, while G_{ui} is an undirected and static graph. It is a big difference in graph types. It is a challenge to learn useful features, so we design the dual-path GCN to tackle the two kinds of graphs separately. Besides, different normalization parameters are designed for them.

(2) The different meanings in reality: G_u and G_i are built based on similarity, while G_{ui} is the interaction records in reality. G_u and G_i are homogeneous graph while G_{ui} is a heterogeneous graph, they should be treated differently for the difference influence among nodes. For example, the influence of one user's friend on him may be more effective than items that he has interacted with when to make the decision to watch a movie or listen to a song.

(3) Quality difference: The edges in G_{ui} is the real interactions as mentioned before, while the links in G_u and G_i are generated, approximately. The quality of the two new graphs is also closely related to the dataset and the performance of the whole algorithm. This design can also avoid mutual influence between the different quality of the graphs.

Besides, the weights in graphs are used to calculate normalization constant $\mathcal{N}$ in the GCN process. We use symmetric normalization, $\frac{1}{\sqrt{|\mathcal{N}_u\mathcal{N}_i|}}$, for the undirected symmetric graphs, G_{ui}, and left normalization, $\frac{1}{|\mathcal{N}_{u'}|}$, $\frac{1}{|\mathcal{N}_{i'}|}$, for directed asymmetric graphs G_u and G_i. Graphs are generated based on similarity matrices row by row, while the left normalization is the row normalization on matrices, so they are matched. The left normalization is asymmetric normalization which is more suitable for directed graphs derived from asymmetric similarity matrices (the bold part in the similarity matrix in Fig. 2).

3.3.2. Message aggregation

In the message aggregation stage, the previous messages are used to enrich the user's or item's representations. A self-loop to keep and learn features of itself is always added in GCN methods, that is:

$$m_u = W_1 e_u \tag{9}$$

$$m_i = W_1 e_i \tag{10}$$

The aggregation functions contain summation, average, maximum, the commonly used summation function is adopted here. The message aggregation is defined as:

$$e_u = \sigma\Big(\sum_{i\in\mathcal{N}_u} m_{u\leftarrow i} + \sum_{u\in\mathcal{N}_{u'}} m_{u\leftarrow u'} + m_u\Big) \tag{11}$$

$$e_i = \sigma\Big(\sum_{i\in\mathcal{N}_i} m_{i\leftarrow u} + \sum_{i\in\mathcal{N}_{i'}} m_{i\leftarrow i'} + m_i\Big) \tag{12}$$

where σ is the activation function, and Relu is used as the activation function in this paper.

If more than one GCN layers are used, the message construction and aggregation mechanism can be described as follows:

$$\begin{aligned} e_u^{(k+1)} = \sigma\Big(&\sum_{i\in\mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u\mathcal{N}_i|}}(W_1 e_i^{(k)} + W_2(e_i^{(k)} \odot e_u^{(k)}))+ \\ &\sum_{i\in\mathcal{N}_u} \frac{1}{|\mathcal{N}_{u'}|}(W_3 e_{u'}^{(k)} + W_4(e_{u'}^{(k)} \odot e_u^{(k)})) + W_1 e_u^{(k)}\Big) \end{aligned} \tag{13}$$

$$\begin{aligned} e_i^{(k+1)} = \sigma\Big(&\sum_{u\in\mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u\mathcal{N}_i|}}(W_1 e_u^{(k)} + W_2(e_u^{(k)} \odot e_i^{(k)}))+ \\ &\sum_{u\in\mathcal{N}_i} \frac{1}{|\mathcal{N}_{i'}|}(W_3 e_{i'}^{(k)} + W_4(e_{i'}^{(k)} \odot e_i^{(k)})) + W_1 e_i^{(k)}\Big) \end{aligned} \tag{14}$$

where k denotes the kth GCN layer.

3.4. Dynamic evolution mechanism

Our dynamic evolution mechanism is the core of this method, including dual-path GCN, final embeddings, graphs construction and reuse, as well as the update and co-evolution of embeddings and graphs. Next, we will introduce the parts except the GCN layers in detail.

3.4.1. Final embeddings

After propagating by GCN layers, we get multiple representations of users and items. With the neighborhood aggregation mechanism, the more GCN propagation layers the farther neighbors' information is aggregated. To make the full use of initial embeddings and representations from different GNN propagation layers, we concatenate them as the final embeddings following the NGCF's approach:

$$E_u = [e_u^0,\ e_u^1, e_u^2, \ldots, e_u^k] \tag{15}$$

$$E_i = [e_i^0,\ e_i^1, e_i^2, \ldots, e_i^k] \tag{16}$$

where e_u^0, e_i^0 are the initial embeddings of users and items, e_u^1, e_u^2, $\ldots$, e_u^k and e_i^0, e_i^1, e_i^2, $\ldots$, e_i^k are the 1, 2, ..., kth output of GCN layers, [ ] denotes the concatenation operation.

3.4.2. Graphs construction and reuse

How to build user and item graphs are described here and the user graph is taken as an example. E_u is used to calculate similarity by the cosine similarity and build a similarity matrix for users. The cosine value is recorded as the edge weight. Then we choose the top k close users as neighbors in each row and establish directed links from neighbors to the user of the row. In this way, we form a directed and weighted graph G_u for all users. G_i is constructed in the same way.

We take the user graph as an example to illustrate why the two generated graphs are directed graphs. We choose the closest neighbor for a user to build the user graph as shown in Fig. 2. Suppose that the four users with blue, green, gray, and yellow colors represent u_1, u_2, u_3, u_4, respectively. The user similarity results are shown in the left of the "Similarity Matrices", and the established user relationship, G_u, is shown on the left of "Multiple Graphs". Similarity values are recorded as the weight of the graph, and the similarity matrix is symmetric. The nearest user is calculated by ranking in a row for each user and shown in bold. The bold values in the similarity matrix are asymmetric, so the G_u is a directed graph. u_1 and u_2 is two-way link. u_1 is the most similar to u_3, but u_3 is not the most similar to u_1. So a one-way connection to u_1 is established between them. Therefore, the user graph is directed and weighted. Moreover, this is in line with the social network in the real world for the social graph can be directed and weighted.

The generated graphs are used as side information in latent space to form a multi-graph collaboration mechanism. The graph G_u with links between similar users simulates the social network. Since different items may belong to the same category and share the same attributes, connections can be mined naturally by our item graph. The graph G_i simulates the knowledge graph from this perspective. Thus, both of them simulate and replace the side information in the real-world and used as collaborative signals for a better rating prediction with the original interaction graph G_{ui}.

3.4.3. Continuous updating for dynamic evolution

Embeddings and the generated graphs need a multi-step optimization process according to deep learning optimization theory. The multiple graphs and embeddings promote each other in our model. For each epoch, G_u and G_i are recalculated and updated, so that it is easy to use the latest and optimized embeddings to obtain better G_u and G_i. At the same time, new G_u and G_i are reused by the next round of training. The better G_u and G_i information are injected into the network to promote the embeddings and achieve high-quality collaborative filtering by the GCN layers. Thus both the multi-graph and embeddings should be continuously updated, to be better step by step, forming a dynamic evolution mechanism. It should be pointed out that G_{ui} is static while G_u and G_i evolve in the training process in our model. Without any side information, it is also a self-evolving mechanism by mining its relationships from scratch.

3.5. MLP Layer

It is insufficient for modeling the collaborative filtering effect just by an element-wise inner product, especially in this dynamic, multi-graph framework. To address this issue, we combine the features of users and items by concatenating them, then one MLP layer is used. The MLP makes interactions between users' and items' latent features and implements the transformation from the final embeddings to the rating score.

$$h_1 = [E_u, E_v] \tag{17}$$

$$h_2 = \sigma(W_{h1}h_1 + b_{h1}) \tag{18}$$

$$h_3 = \sigma(W_{h2}h_2 + b_{h2}) \tag{19}$$

$$\hat{R}_{ui} = W_{h3}h_3 + b_{h3} \tag{20}$$

where h_1, h_2, h_3 denote the features, W_{h1}, W_{h2}, W_{h3} are the weight matrix, b_{h1}, b_{h2}, b_{h3} denote bias vectors, the activation function σ is Relu, $\hat{R}_{u,i}$ is the predicted rating score.

We empirically implement the tower structure for the first two layers, halving the layer size for each successive higher layer, such as [64, 32]. By using a small number of hidden units for higher layers, they can learn more abstractive features of data. At last, we get only one output, namely, the prediction. It is worthwhile to mention that NGCF adopts the same MLP for the rating prediction task for a fair comparison with us.

3.6. Loss function

We use the loss function commonly used in rating prediction by minimizing deviation between the predictions and the ground truth ratings, and regularization is added to prevent overfitting:

$$\text{Loss} = \frac{1}{\Omega}\sum_{(u,i)\in\Omega}(\hat{R}_{u,i} - R_{u,i})^2 + \lambda\,||\Theta||_2^2 \tag{21}$$

where Ω denotes the edges or links in training data, $R_{u,i}$, $\hat{R}_{u,i}$ denote the ground truth and predicted score, Θ denotes all trainable model parameters, λ trades-off the importance of the two loss functions.

3.7. Model analysis

Model Size: The model parameters contain three parts, initial embeddings, GCN layers and the MLP layer. The total size is $(m+n)d+\sum_{l=1}^{L} 4d_l d_{l-1}+\sum_{h=1}^{H} d_h d_{h-1}$, where m, n denote the user and item numbers; L, H denote the number of GCN layers and MLP hidden layers, respectively; d, d_l, d_h denote the dimension of parameters of the three parts. The difference between DMGCF and NGCF in model size is the new GCN path which brings $\sum_{1=1}^{L} 2d_1 d_{1-1}$ more parameters. Considering that d_l is always set to 32 or 64, L is usually a number smaller than 5, which is much smaller than m, n. Thus, the total parameters increased are quite limited. For example, if we set L=2, d_l=32, the model size of NGCF of Flixster, Yelp are about 202K, 1314K respectively, while our DMGCF uses only 4K more parameters. To summarize, DMGCF can achieve better results than NGCF with almost the same model size.

Time Complexity Analysis: There computational complexity of DMGCF contains the dual-path GCN, the cosine similarity and sorting for top-k links to build graphs, and the MLP layer. For the dual-path GCN, the computational complexity of the

lth propagation layer is $O(\sum_{l=1}^{L}\left(\left|\mathcal{R}^{+}+k(m+n)\right|\right)d_l d_{l-1})$, where $\mathcal{R}^{+}$, $k(m+n)$ denote the number of nonzero values of the user-item graph, the generated user and item graphs, respectively; k denotes the number of links added for each user and item. $O(mn\sum_{l=0}^{L}d_l)$, $O(m\log m+n\log n)$ are computational complexities for cosine similarity and sorting for top-k relations. The computational complexity of MLP is $O(\sum_{h=1}^{H}\left|\mathcal{R}^{+}\right|d_h d_{h-1})$. Thus, the total computational complexity of DMGCF is $O(\sum_{l=1}^{L}(\left|\mathcal{R}^{+}\right|+k(m+n))d_l d_{l-1}+mn\sum_{l=0}^{L}d_l+m\log m+n\log n+\sum_{h=1}^{H}\left|\mathcal{R}^{+}\right|d_h d_{h-1})$, while that of NGCF is $O(\sum_{l=1}^{L}\left|\mathcal{R}^{+}\right|d_l d_{l-1}+\sum_{h=1}^{H}\left|\mathcal{R}^{+}\right|d_h d_{h-1})$. The difference lies in the construction and calculation of the added graphs. According to the subsequent experimental analysis in Section 4, we can reduce the time complexity by using only the item graph and reducing the number of k.

Empirically, d_l, d_h are always set to 32, 64, which are smaller. $\left|\mathcal{R}^{+}\right|$, m, n of the recommendation system are large, which are the main factor affecting the computational complexity. Under the same experimental settings (as explained in Section 4) on the GeForce RTX 2080Ti GPU, NGCF costs about 2.3s, 16.3s, 4.5s per epoch on ML-100K, ML-1M, Yelp, while DMGCF costs 2.6s, 17.4s, 10 s for training, respectively. There is almost no time difference in test for both NGCF and DMGCF cost about 0.2s, 1.2s, 0.3s on the three datasets. Because there is no need to build graphs when test, and the added graphs are also very sparse, the computation can be accelerated by the GPU. The above results showed that the time costs are acceptable and m, n have a bigger impact on train time. DMGCF has more advantages in the application for having almost the same test time as NGCF.

4. Experiments

In this section, a series of experiments are conducted to study the proposed model. First, the basic information of the experiments is introduced, such as datasets, performance measures, experimental details, compared methods. Then experimental results, ablation study, and many discussions are elaborated.

4.1. Datasets

We evaluate our model on five commonly used rating perdition benchmark datasets:

Flixster [57]: Flixster is a movie rating dataset obtained from the Flixster website. The scale of the rating is 0.5–5, and the interval between the ratings is 0.5. Following sRMGCNN [54], 3000 users and items are used for experiments.

YahooMusic [58]: This dataset is released based on Yahoo! Music ratings through the KDD Cup 2011 contest. The sparsity of the dataset is relatively high when compared to other collaborative filtering datasets. A reduced matrix of 3000 users and items following sRMGCNN.

MovieLens 100K (ML-100K), MovieLens 1M (ML-1M) [59]: MovieLens datasets are widely used for rating prediction and recommendation. They contain the rating data of users for multiple movies from the MovieLens website with timestamps. There are several versions for different amounts of data, we use the size of 100K and 1M. For ML-100K, the first of the 5 provided data splits is used.

Yelp[1]: Yelp dataset is adopted from the 2020 edition of the Yelp challenge from America's biggest review site. Local businesses like restaurants are viewed as items. We random sample 15% of rating records and select users and items having at least 5 ratings. The dataset is sparser and the number of users and items are larger than the other datasets.

[1] https://www.yelp.com/dataset

Table 1
Detailed information for datasets.

Dataset	Users	Items	Ratings	Density	Rating types
Flixster	3000	3000	26173	0.0029	0.5, 1, ..., 5
YahooMusic	3000	3000	5335	0.0006	1, 2, ..., 100
ML-100K	943	1682	100000	0.063	1, 2, 3, 4, 5
ML-1M	6040	3706	1000209	0.0447	1, 2, 3, 4, 5
Yelp	16793	23953	172567	0.0004	1, 2, 3, 4, 5

The statistics of these datasets are shown in Table 1. We completely followed the dataset setup provided by sRMGCNN [54] in which the train, test datasets are split already[2] for the first three datasets. They are different in numbers of ratings, density, and rating types. The MovieLens 1M dataset is used because the rating numbers are large. Yelp is used because its user and item numbers are large and the dataset is sparser. The train set is 90% of the whole dataset besides ML-100K accounts for 80%. We randomly select 10% of the train set as validation set to tune hyperparameters.

4.2. Performance measures

The most popular accuracy measurements used in rating prediction are Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) [28,39,40], which are defined as follows:

$$\mathrm{RMSE}=\sqrt{\sum_{i\in N_{test}}\left(\hat{R}_{u,i}-R_{u,i}\right)^2/|N_{test}|} \tag{22}$$

$$\mathrm{MAE}=\sum_{i\in N_{test}}\left|\hat{R}_{u,i}-R_{u,i}\right|/|N_{test}| \tag{23}$$

where $R_{u,i}$, $\hat{R}_{u,i}$ denote the ground truth and predicted rating score, and N_{test} denotes the number of user-item pairs in the test set.

4.3. Experimental details

We implement our model in Pytorch[3] to take advantage of its dynamic graph mechanism[4]. The default embedding size is 32 for all methods, and one GCN layer is used in our model. We optimize all models with the Adam optimizer, the batch size is fixed at 2048 except that of ML-1M is 10240. The learning rate is tuned amongst [0.0001, 0.0005, 0.001, 0.005] , the coefficient of L2 normalization is searched in [0.0001, 0.001, 0.01, 0.1, 1]. We do not use node dropout strategy employed by NGCF. If there is no special explanation, we generally establish 5 links for each user and item in the generated graphs. Kaiming initializer is used to initialize the model parameters.

4.4. Compared methods

In addition to the traditional matrix factorization method SVD [14], we select four state-of-the-art recommenders as the competitors, NCF [22], DeepFM [60], sRMGCNN [54], NGCF [10].

SVD [14]: This model is a classical rating prediction method in the matrix factorization approach with user and item bias to model the preference or difference of users and items.

NCF [22]: He et al. propose three neural collaborative filtering (NCF) methods in a deep learning way. The best model

[2] https://github.com/fmonti/mgcnn

[3] https://pytorch.org

[4] https://github.com/th971286733/DMGCF

Table 2
Performance comparison.

	Flixster		YahooMusic		ML-100K		ML-1M		Yelp	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
SVD [14]	0.9476	0.7486	54.71	47.69	0.9856	0.7922	0.9850	0.8014	1.1250	0.9064
NCF [22]	0.9150	0.7076	26.44	21.17	0.9390	0.7415	0.8692	0.6820	1.0889	0.8630
DeepFM [60]	0.9109	0.7001	25.47	20.33	0.9489	0.7531	0.8834	0.6943	1.0926	0.8610
sRMGCNN [54]	0.9261	0.7139	22.31	18.76	0.9292	0.7378	0.8776	0.6920	1.0881	0.8719
NGCF [10]	0.9098	0.6888	21.49	17.41	0.9319	0.7321	0.8670	0.6802	1.0931	0.8567
DMGCF(ours)	**0.8928**	**0.6691**	**20.55**	**16.10**	**0.9234**	**0.7254**	**0.8586**	**0.6724**	**1.0801**	**0.8519**
Improv.	1.87%	2.86%	4.37%	7.52%	0.62%	0.92%	0.97%	1.15%	0.74%	0.56%
p-value	8.22E−06	2.84E−06	2.10E−06	2.24E−06	1.05E−03	3.30E−04	2.81E−05	5.00E−05	5.41E−03	1.18E−01

The best and the second-best results are in bold or underlined respectively. "Improv." indicates the improvements of DMGCF over the second-best results.

NeuMF is adopted here. It combines traditional matrix factorization and deep neural network, which can simultaneously extract low and high dimensional features. It contains two submodules, a generalized matrix factorization (GMF) model to realize matrix decomposition and a MLP to learn deep and nonlinear relationships.

DeepFM [60]: DeepFM is one of the well-known and widely used recommenders.DeepFM combines the power of factorization machines for recommendation and deep learning for feature learning in a new neural network architecture. It introduces a sharing strategy of feature embedding to avoid feature engineering. DeepFM makes it possible to derive an end-to-end learning model that emphasizes both low-order and high-order feature interactions.

sRMGCNN [54]: sRMGCNN solves the rating prediction problem in a matrix completion way. A deep multi-graph CNN framework using Chebychev polynomial filters is proposed to extract meaningful statistical patterns on two graphs, user graph and item graph. It is worth noting that the user and item graphs built in the space of user and movie features or from the scores of the original matrix are used in sRMGCNN. Thus, it makes a good comparison with us for sRMGCNN with side information and multiple graphs.

NGCF [10]: NGCF is the state-of-the-art model of the graph-based CF method. NGCF exploits high-order connectivity in the user-item graph by propagating embeddings on it, effectively injecting the collaborative signal into the embedding process in an explicit manner. We keep the same settings for a fair comparison with our model.

4.5. Performance comparison

The experimental results are shown in Table 2 best results are shown in bold, and the second-best results are underlined. Our model achieves the best results compared to all the comparison methods.

As a basic matrix factorization method, the performances of SVD are poor. NCF and DeepFM get better results than SVD, which shows the advantages of deep learning over the traditional method. In general, the graph-based approaches (sRMGCNN, NGCF, and DMGCF) are better than the deep learning methods without graphs (NCF and DeepFM), which demonstrates the importance of graphs and graph convolution network. Most results of NGCF on these datasets have been improved compared with sRMGCNN, for sRMGCNN is the spectral GNN method based on the user graph and item graph which are built by design while NGCF is based on the advanced GCN method in the spatial domain on the real user-item graph.

DMGCF consistently gains the best on all the datasets. From the comparison of the five methods, DMGCF improves over the second-best method NGCF by 1.87%, 4.37%, 0.91%, 0.97%, 1.19% on the five datasets with respect to RMSE. The improvements are significant, even though NGCF is the best graph-based CF method at present. And the improvements relative to the second-best results are shown in Table 2 in the row of "Improv". The results show that DMGCF improves significantly on YahooMusic and Flixster, and owns about a 1% increase on ML-1M, ML-100k, and a slight increase on Yelp. We conduct one-sample t-tests and p-value <0.05 indicates that the improvements of DMGCF over the second-best results are statistically significant only except MAE on the Yelp dataset. Besides, compared with sRMGCNN which uses side information and multiple graphs, the improvement of DMGCF is more obvious. Thus, these performances and comparisons show the advancement of our model.

Besides, we observe that the improvement of MAE is always better than that of the RMSE. The differences between MAE and RMSE are very close, and the MAE value is smaller than the RMSE in general, so the improvements on MAE is larger than RMSE here.

4.6. Ablation study

It is important to explore whether our design works. The ablation study is divided into four parts. The effects of multiple graphs and parameters we designed are discussed in the first two parts. Then, just adding G_u or G_i as variants of DMGCF are discussed. Besides, the comparison of dynamic learning mechanism and static learning are compared. The whole results are shown in Table 3. The ID here is the number of the different methods. Method ID 6 is our DMGCF. It is important to note that multiple graphs are generated dynamically, so the dynamic evolution mechanism is added when we add generated graph unless it is removed when we study the effect of the dynamic mechanism alone (method 5).

4.6.1. Effect of the added multiple graphs

In order to show just the G_u, G_i are added without more parameters, we need additional design to form a variant of the model. For message construction, the parameters W_3, W_4 are reduced by treating message passing and affinity message as two types of signals, that is to say, $W_3 = W_1$, $W_4 = W_2$, the added path is removed. In this case, the user graph and item graph are treated the same as the original graph for simple. We show the formula clearly:

$$e_u^{(k+1)} = \sigma\Big(\sum_{i\in\mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_{uu'}\mathcal{N}_{ii'}|}}(W_1(e_i^{(k)} + e_{u'}^{(k)}) + W_2(e_i^{(k)} \odot e_u^{(k)} + e_{u'}^{(k)} \odot e_u^{(k)})) + W_1 e_u^{(k)}\Big) \tag{24}$$

$$e_i^{(k+1)} = \sigma\Big(\sum_{u\in\mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_{uu'}\mathcal{N}_{ii'}|}}(W_1(e_u^{(k)} + e_{i'}^{(k)}) + W_2(e_u^{(k)} \odot e_i^{(k)} + e_{i'}^{(k)} \odot e_i^{(k)})) + W_1 e_i^{(k)}\Big) \tag{25}$$

where the $\mathcal{N}_{uu'}, \mathcal{N}_{ii'}$ contains links within two graphs.

Table 3
Results of ablation study.

	COMPONENTS					Flixster		YahooMusic		ML-100K		ML-1M		Yelp	
ID	NGCF	Dy	G_u	G_i	Para	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
1	✓					0.9088	0.6888	21.49	17.41	0.9319	0.7321	0.8670	0.6802	1.0931	0.8567
2	✓	✓	✓	✓		0.9020	0.6813	21.21	17.02	0.9249	0.7277	0.8673	0.6821	1.0917	0.8534
3	✓	✓	✓		✓	0.9059	0.6846	21.06	16.54	0.9254	0.7278	0.8593	0.6733	1.0840	0.8525
4	✓	✓		✓	✓	**0.8906**	**0.6743**	20.59	16.13	0.9239	0.7254	**0.8586**	**0.6724**	1.0814	0.8523
5	✓		✓	✓	✓	0.8967	0.6771	21.15	16.96	0.9255	0.7277	0.8613	0.6758	1.0982	0.8587
6	✓	✓	✓	✓	✓	0.8931	0.6753	**20.56**	**16.10**	**0.9238**	**0.7249**	0.8605	0.6735	**1.0814**	**0.8518**

where 'ID', 'Dy', 'Para' refer to the ID of methods, the dynamic evolution mechanism, the added parameters, respectively. Method 6 is ours DMGCF. The dynamic evolution mechanism is used simultaneously with multiple graphs, except when its effects are studied separately (method 5).

This variant has the same parameters as NGCF and is shown in ID 2 in Table 3. The results of ID 1 and 2 show that the overall trends of RMSE and MAE become better with the addition of G_u, G_i in the first three datasets, while there are no significant changes in the two large datasets. This illustrates the effectiveness of the multi-graph on small datasets but still facing challenges on big datasets. More design is needed for multi-graph collaborative filtering, as can be seen below.

4.6.2. Effect of the added parameters

The results are listed in IDs of 2, 6 in Table 3. Adding parameters based on method 2, the RMSEs and MAEs are improved obviously except Yelp with a slight improvement of MAE. The relative improvement ratio of RMSE are 1.73%, 4.33%, 0.869%, 0.749%, 1.07%, that of MAE are 1.95%, 7.52%, 0.983%, 0.985%, 0.258%. YahooMusic and Flixster datasets benefit most. Results show the effectiveness of the new path with parameters we designed for different multiple graphs.

4.6.3. Variants of DMGCF, DMGCF-U and DMGCF-I

It is necessary to discuss the specific situation of adding only G_u or G_i, termed DMGCF-U and DMGCF-I respectively. They can be seen as variants of DMGCF. Methods 3, 4, 6 in Table 3 make it clear for contrast. Results show that DMGCF-U, DMGCF-I, and DMGCF are all better than the original NGCF. Generally, DMGCF-I has a significant improvement than DMGCF-U, and DMGCF-I on Flixster and ML-1M achieves the best results. DMGCF gets the best results in the other three datasets. Nevertheless, the best results of the other three datasets in DMGCF and DMGCF-I are also very close.

It is interesting to find that DMGCF-I is better than DMGCF-U, or even gets the best results sometimes. Similar to us, AutoRec [20] also found that the item-based method is more efficient than the user-based. It makes sense for the following reasons. On one hand, user graph-based CF tends to focus on the common preferences of users in their interest groups and pays more attention to socialization, while item graph-based CF recommends similar items based on users' historical behaviors and pays more attention to personalization. The user's rating behavior is more personalized, thus the item graph works better for movies, music, and book sites. On the other hand, users always have a large range of personal preferences and some users may give noise ratings while items are more objective. The connections established by items are relatively good by their fixed attributes and categories. For example, in music, movie and e-commerce websites, their tree-like classification is an objective and real connection, which can be mined through embedding features. Thus, adding G_u and G_i at the same time usually has both the advantages of collaboration and the difficulty of noise. However, it is worth noting that both DMGCF and variants are obviously improved compared to NGCF. This also suggests that DMGCF-I can be used for simplicity.

Table 4
The impact of embedding size of dual-path GCN.

Embedding size	Flixster		YahooMusic		ML-100K	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
16	0.9105	0.6949	20.9344	16.8077	0.9266	0.7274
32	**0.8926**	**0.6736**	**20.6872**	**16.3879**	**0.9238**	**0.7249**
64	0.8966	0.6776	20.9662	16.3886	0.9286	0.7294
128	0.9088	0.6772	21.8890	17.1163	0.9546	0.7547

Table 5
The impact of parameters W_1- W_4 of dual-path GCN.

W_size	Flixster		YahooMusic		ML-100K	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
16	0.9051	0.6867	21.3959	17.0251	0.9267	0.7267
32	**0.8926**	**0.6736**	20.6872	16.3879	**0.9238**	**0.7249**
64	0.9011	0.6764	20.6503	16.2061	0.9266	0.7266
128	0.8928	0.6762	**20.1420**	**15.6566**	0.9529	0.7528

4.6.4. Effect of dynamic graphs

Here, we illustrate the advantages of dynamic graphs over static graphs. A method based on static graphs is designed for comparison. We first calculate the cosine similarity between users (or items) based on the rating matrix, and then select the top-n users (items) to establish G_u and G_i as our DMGCN does. Then we use the same formulas in Section 3. The difference compared with the dynamic approach is that user graph and item graph are calculated at the beginning by rating matrix and remain unchanged while the dynamic way is to dynamically build, use and update these two graphs by embeddings throughout the training process. Methods 5 and 6 in Table 3 show that the dynamic mode achieves better results than the static one, especially on YahooMusic and Yelp datasets. RMSE and MAE of Yelp are both the worst without dynamic graphs, while the best result is gained when adding it.

4.7. Discussion

In this part, the dual-path GCN, the added graphs, and the dynamic evolution mechanism are discussed in detail for a better understanding of the model. The first three datasets are used here for they represent three different levels of sparsity and have three different rating types, a 5-point scale with 1 or 0.5 interval, a 100-point scale. Different rating types take into account the wide range of practical applications and also increase the difficulty of rating prediction.

4.7.1. Discussion of the parameters of the dual-path GCN

The impact of embedding size of dual-path GCN. Four different embedding sizes, [16, 32, 64, 128], are considered. The results are shown in Table 4. As the size increases, both RMSE and MAE decrease first and then increase. The size 16 is too small to reflect the difference between items or users. The results

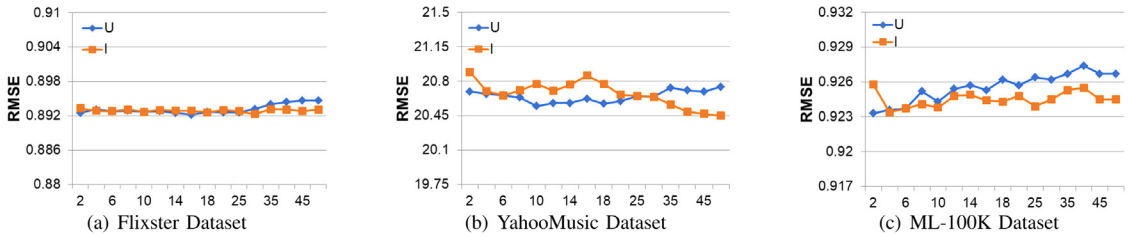


Fig. 3. The impact of the number of links in user graph and item graph. The horizontal axis represents the number of links added. We set the NGCF result as the maximum value of the vertical coordinate to observe the trend.

Table 6
The impact of the layer number of dual-path GCN.

Layer number	Flixster		YahooMusic		ML-100K	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
1	**0.8926**	**0.6736**	20.6872	16.3879	0.9238	**0.7249**
2	0.8934	0.6785	**20.3458**	**16.0628**	0.9250	0.7256
3	0.9012	0.6822	21.1385	16.0707	**0.9236**	0.7253
4	0.9022	0.6904	21.1069	16.0759	0.9251	0.7256

Table 7
The impact of the weighted graphs.

Weighted graph	Flixster		YahooMusic		ML-100K	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
Yes	**0.8926**	**0.6736**	**20.6872**	**16.3879**	**0.9238**	**0.7249**
No	0.8998	0.6879	20.6928	16.3979	0.9246	0.7281

are significantly worse at 128 because the embedding size is too large, which makes the learning process more difficult and increases the risk of over-fitting. This experiment shows that choosing the right size is helpful to improve the effect, and the size 32 is the best one.

The impact of parameters size of dual-path GCN. W_1-W_4 are parameters of dual-path GCN. They are the same size of [embedding size, output size] in our model. The output size of them is adjusted to [16, 32, 64, 128] to observe the effect. The results of the three datasets are shown in Table 5. It can be seen that 32 is appropriate for Flixster and ML-100K dataset, and 128 is better for YahooMusic. Thus, our default setting of 32 is reasonable. YahooMusic performs better with a large parameter size may because it has more complex rating types, which requires more parameters.

The impact of the layer number of dual-path GCN. Table 6 shows the impact of the layer number of dual-path GCN. Flixster, YahooMusic achieve the best results on one or two layers, and the 3 or 4 layers became worse significantly. The results of ML-100K fluctuate. Considering the two accuracy measurements, one layer is better. At present, the common algorithms based on graph convolution generally use few layers, because the aggregation of more layers will make the representation of nodes tend to be smooth, and may introduce more noises. The connections we established have increased the number of nodes that need to be aggregated. Therefore, few layers or just one layer are suitable for our method.

4.7.2. Discussion of the added graphs

The impact of the weighted graphs. The user and item graphs designed in this paper are weighted graphs, and the weights are used in the GCN layers. To investigate whether DMGCF can benefit from the weighted graphs, we compare it with the unweighted graphs where only 0 and 1 to indicate whether an edge has been established. The results in Table 7 show that the weighted graphs are better than the unweighted graphs, which verify the effectiveness of weights in the generated graphs.

The impact of the number of links in user graph and item graph. Though the number is a hyperparameter in experiments, we find that different datasets have different performances, and there are still rules on the same dataset. We use the variable control method to study the influence of the change of the number of one factor, and the range of number is [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 25, 30, 35, 40, 45, 50], which begins with a compact and detailed scope and ends with a large scope. If the vertical coordinate range in the figure is small, the curve will fluctuate greatly. Therefore, we set the NGCF result as the maximum value of the vertical coordinate to discuss the data fluctuation from a proper angle. The examples of default number 5 for the fixed factor are representative after many experiments, and the RMSE results are shown in Fig. 3. Results show that the Flixster dataset has a small fluctuation range and is almost stable. As the number increases, the RMSE starts to deteriorate when the number is greater than 35. On the YahooMusic dataset, RMSE moves up and down slowly, and the fluctuation range of the item is larger than that of the user. For The ML-100K dataset, both user and item own a rising trend with the increase of the number, and the fluctuation range is between 0.923 and 0.927. Compared with the NGCF, the figure shows that a smaller number of links are acceptable. It is not a difficult task to adjust this parameters, 5 links in our experiments can be effective.

4.7.3. Discussion of the dynamic evolution mechanism

The impact of the start time of the dynamic evolution mechanism. When to start the dynamic evolution mechanism is another important question. From 0.1, 0.2, 0.3, …, 0.9 times the total number of training epochs, we start the dynamic evolution mechanism and the collaborative evolution of multiple graphs and embeddings. For example, we start to build and update the user graph and item graph at 0.1 of the total train epochs and run the collaborative evolution of multiple graphs and embeddings for all the rest epochs.

We show RMSE and MAE results of three datasets in Fig. 4. The common trend of RMSE and MAE is that they decrease first and then increase as the starting point changes. The YahooMusic and Flixster dataset have a trend of the checkmark and long tail with a slight and reasonable fluctuation, while the ML-100K dataset is concave and fluctuates greatly. Generally speaking, the result of starting the dynamic mechanism in the initial stage is not optimal. Starting the dynamic mechanism after training for some time can get the best results. When starting too late the result becomes worse gradually.

This phenomenon is reasonable and in line with expectations. The result of starting the dynamic update mechanism at the beginning is not optimal. Because the embedding has not been improved and trained sufficiently, and the quality of them is still poor, so are the graphs based on embeddings. With the advancement of training time, the dynamic mechanism is started at a

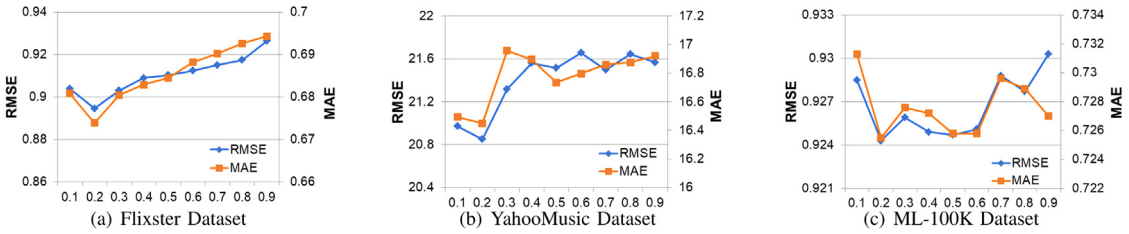


Fig. 4. The impact of start time of dynamic evolution mechanism. The horizontal axis represents the start time of the whole epochs. RMSE and MAE have a different range and are represented separately by the left and right vertical axes.

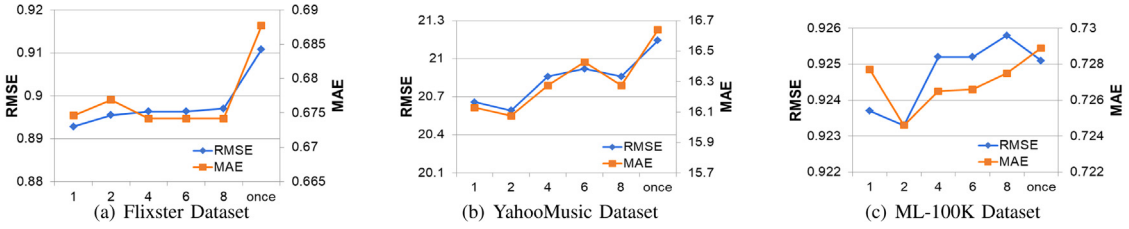


Fig. 5. The impact of update frequency and the dynamic evolution mechanism. The horizontal axis represents the update interval, and 'once' means multiple graphs build only once without any update. RMSE and MAE have a different range and are represented separately by the left and right vertical axes.

reasonable point when the embedding quality is guaranteed, and the higher quality of the graph is established. The co-evolution mechanism we expect is well established and the best results are obtained. As time goes on, RMSE and MAE curves increase gradually in the middle and late stages of training, and the results become worse and worse. Due to the existence of overfitting, over-learning of embedding parameters led to a poor graph that established and updated at this time. Therefore, the right startup time is an important issue.

The conclusion when the dynamic evolution mechanism begins can be drawn from Fig. 4. The best result can be achieved by initiating the dynamic evolution mechanism during a short period of initial training, about 20% of the total epochs. The training epochs are based on the dataset and our training epochs are the default settings.

The impact of update frequency and the dynamic evolution mechanism. How often is better to update the user graph and item graph? Each epoch updated in our default settings may not be the best. It is necessary to explore the update frequency. We start the dynamic evolution mechanism at 0.2 of the whole epoch and update after every 1, 2, 4, 6, 8 epochs, and 'once' means multiple graphs build only once at 0.2 times epoch and without any update, which is equivalent to a very large update interval.

It can be seen from Fig. 5 that the trends of RMSE and MAE are relatively consistent, and RMSE is used as the main indicator to analyze. Experimental results show that it is better to update with smaller intervals. The best RMSE for Flixster, YahooMusic, and ML-100K dataset is updated every 1, 2, and 2 epochs respectively. RMSE of the Flixster dataset changes relatively smoothly as the update interval increases, while YahooMusic and ML-100K dataset get worse greatly. It is worth noting that the 'once' is very poor for all datasets. Impact of update frequency strongly prove the necessity and the important role of the dynamic evolution mechanism.

5. Conclusion

In this work, we propose the model of DMGCF based on multiple graphs and dynamic evolution mechanism. The idea of mining relations and generating multiple graphs to simulate side information for better performance is feasible. Our DMGCF designed for different types of multiple graphs is effective. It is generally effective to add the generated multiple graphs. All results get better when adding a new path with parameters which can deal with multiple graphs with different categories. Dynamic evolution mechanism is necessary and it is better than the method with a static graph. The discussion also shows the superiority of the model.

For the collaborative filtering models based on multiple graphs, DMGCF can be seen as the beginning of the automatic generation of graphs and dynamic evolution mechanism. This work can be expanded to many other challenging and excellent works such as adding advanced methods for graph generation or more suitable losses. Our work also provides useful references in other areas for graph generation, dynamic graphs, and multi-graph based collaborative filtering of the recommendation system.

CRediT authorship contribution statement

Hao Tang: Methodology, Software, Data curation, Writing - original draft, Formal analysis. **Guoshuai Zhao:** Conceptualization, Methodology, Formal analysis, Supervision. **Xuxiao Bu:** Writing - review & editing. **Xueming Qian:** Resources, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the NSFC, China under Grants 61902309, 61701391, and 61772407; in part by ShaanXi Province under Grant 2018JM6092; in part by the Fundamental Research Funds for the Central Universities, China (xxj022019003); in part by China Postdoctoral Science Foundation (2020M683496); and in part by the National Postdoctoral Innovative Talents Support Program, China (BX20190273).

References

[1] R. Chen, Q. Hua, Y. Chang, B. Wang, L. Zhang, X. Kong, A survey of collaborative filtering-based recommender systems: From traditional methods to hybrid methods based on social networks, IEEE Access 6 (2018) 64301–64320.

[2] Y. Shi, M.A. Larson, A. Hanjalic, Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges, ACM Comput. Surv. 47 (1) (2014) 3:1–3:45.

[3] Z. Sun, Q. Guo, J. Yang, H. Fang, G. Guo, J. Zhang, R. Burke, Research commentary on recommendations with side information: A survey and research directions, Electron. Commer. Res. Appl. 37 (2019).

[4] X. Wang, X. He, Y. Cao, M. Liu, T. Chua, KGAT: knowledge graph attention network for recommendation, in: KDD, ACM, 2019, pp. 950–958.

[5] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, M. Guo, RippleNet: Propagating user preferences on the knowledge graph for recommender systems, in: CIKM, ACM, 2018, pp. 417–426.

[6] H. Wang, M. Zhao, X. Xie, W. Li, M. Guo, Knowledge graph convolutional networks for recommender systems, in: WWW, ACM, 2019, pp. 3307–3313.

[7] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, M. Sun, Graph neural networks: A review of methods and applications, CoRR (2018) abs/1812.08434.

[8] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P.S. Yu, A comprehensive survey on graph neural networks, CoRR (2019) abs/1901.00596.

[9] R. Yin, K. Li, G. Zhang, J. Lu, A deeper graph neural network for recommender systems, Knowl. Based Syst. 185 (2019).

[10] X. Wang, X. He, M. Wang, F. Feng, T. Chua, Neural graph collaborative filtering, in: SIGIR, ACM, 2019, pp. 165–174.

[11] S.R. Gandhi, J. Gheewala, A survey on recommendation system with collaborative filtering using big data, in: 2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), 2017, pp. 457–460.

[12] M. Jalili, S. Ahmadian, M. Izadi, P. Moradi, M. Salehi, Evaluating collaborative filtering recommender algorithms: A survey, IEEE Access 6 (2018) 74003–74024.

[13] Y. Koren, R.M. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, IEEE Comput. 42 (8) (2009) 30–37.

[14] D. Billsus, M.J. Pazzani, Learning collaborative information filters, in: ICML, Morgan Kaufmann, 1998, pp. 46–54.

[15] R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, in: NIPS, Curran Associates, Inc., 2007, pp. 1257–1264.

[16] B. Huang, X. Yan, J. Lin, Collaborative filtering recommendation algorithm based on joint nonnegative matrix factorization, Pattern Recognit. Artif. Intell. (2016).

[17] G. Sun, Y. Cong, Y. Zhang, G. Zhao, Y. Fu, Continual multiview task learning via deep matrix factorization, IEEE Trans. Neural Networks Learn. Syst. 32 (1) (2021) 139–150.

[18] S. Rendle, Factorization machines, in: ICDM, IEEE Computer Society, 2010, pp. 995–1000.

[19] H. Xue, X. Dai, J. Zhang, S. Huang, J. Chen, Deep matrix factorization models for recommender systems, in: IJCAI, ijcai.org, 2017, pp. 3203–3209.

[20] S. Sedhain, A.K. Menon, S. Sanner, L. Xie, AutoRec: Autoencoders meet collaborative filtering, in: WWW, ACM, 2015, pp. 111–112.

[21] Y. Zheng, B. Tang, W. Ding, H. Zhou, A neural autoregressive approach to collaborative filtering, in: ICML, JMLR Workshop and Conference Proceedings, vol. 48, JMLR.org, 2016, pp. 764–773.

[22] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T. Chua, Neural collaborative filtering, in: WWW, ACM, 2017, pp. 173–182.

[23] Z. Sun, Q. Guo, J. Yang, H. Fang, G. Guo, J. Zhang, R. Burke, Research commentary on recommendations with side information: A survey and research directions, Electron. Commer. Res. Appl. 37 (2019).

[24] J. Han, L. Zheng, Y. Xu, B. Zhang, F. Zhuang, P.S. Yu, W. Zuo, Adaptive deep modeling of users and items using side information for recommendation, IEEE Trans. Neural Netw. Learn. Syst. 31 (3) (2020) 737–748.

[25] Y. Chen, X. Zhao, M. de Rijke, Top-N recommendation with high-dimensional side information via locality preserving projection, in: SIGIR, ACM, 2017, pp. 985–988.

[26] F. Zhao, Y. Guo, Learning discriminative recommendation systems with side information, in: IJCAI, ijcai.org, 2017, pp. 3469–3475.

[27] X. Zhang, J. Jia, K. Gao, Y. Zhang, D. Zhang, J. Li, Q. Tian, Trip outfits advisor: Location-oriented clothing recommendation, IEEE Trans. Multimed. 19 (11) (2017) 2533–2544.

[28] X. Lei, X. Qian, G. Zhao, Rating prediction based on social sentiment from textual reviews, IEEE Trans. Multimed. 18 (9) (2016) 1910–1921.

[29] G. Zhao, H. Fu, R. Song, T. Sakai, Z. Chen, X. Xie, X. Qian, Personalized reason generation for explainable song recommendation, ACM Trans. Intell. Syst. Technol. 10 (4) (2019) 41:1–41:21.

[30] G. Zhao, Z. Liu, Y. Chao, X. Qian, Caper: context-aware personalized emoji recommendation, IEEE Transactions on Knowledge and Data Engineering (2020) http://dx.doi.org/10.1109/TKDE.2020.2966971, 1-1.

[31] Y. Wu, K. Li, G. Zhao, X. QIAN, Personalized long- and short-term preference learning for next poi recommendation, IEEE Transactions on Knowledge and Data Engineering (2020) http://dx.doi.org/10.1109/TKDE.2020.3002531, 1-1.

[32] G. Zhao, X. Qian, C. Kang, Service rating prediction by exploring social mobile users' geographical locations, IEEE Trans. Big Data 3 (1) (2017) 67–78.

[33] Z. Cheng, J. Shen, Just-for-Me: An adaptive personalization system for location-aware social music recommendation, in: ICMR, ACM, 2014, p. 185.

[34] G. Zhao, P. Lou, X. Qian, X. Hou, Personalized location recommendation by fusing sentimental and spatial context, Knowl. Based Syst. 196 (2020) 105849.

[35] Z. Cheng, J. Shen, On effective location-aware music recommendation, ACM Trans. Inf. Syst. 34 (2) (2016) 13:1–13:32.

[36] G. Zhao, T. Liu, X. Qian, T. Hou, H. Wang, X. Hou, Z. Li, Location recommendation for enterprises by multi- source urban big data analysis, IEEE Trans. Serv. Comput. 13 (6) (2020) 1115–1127.

[37] D. Cong, Y. Zhao, B. Qin, Y. Han, M. Zhang, A. Liu, N. Chen, Hierarchical attention based neural network for explainable recommendation, in: ICMR, ACM, 2019, pp. 373–381.

[38] C. Chen, M. Zhang, Y. Liu, S. Ma, Neural attentional rating regression with review-level explanations, in: WWW, ACM, 2018, pp. 1583–1592.

[39] J. Wen, J. Ma, H. Tu, M. Zhong, G. Zhang, W. Yin, J. Fang, Hierarchical text interaction for rating prediction, Knowl. Based Syst. 206 (2020) 106344.

[40] J. Zhao, W. Wang, Z. Zhang, Q. Sun, H. Huo, L. Qu, S. Zheng, TrustTF: A tensor factorization model using user trust and implicit feedback for context-aware recommender systems, Knowl. Based Syst. 209 (2020) 106434.

[41] C. Hsu, M. Yeh, S. Lin, A general framework for implicit and explicit social recommendation, IEEE Trans. Knowl. Data Eng. 30 (12) (2018) 2228–2241.

[42] G. Zhao, X. Lei, X. Qian, T. Mei, Exploring users' internal influence from reviews for social recommendation, IEEE Trans. Multim. 21 (3) (2019) 771–781.

[43] G. Zhao, X. Qian, X. Xie, User-service rating prediction by exploring social users' rating behaviors, IEEE Trans. Multim. 18 (3) (2016) 496–506.

[44] Q. Wu, H. Zhang, X. Gao, P. He, P. Weng, H. Gao, G. Chen, Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems, in: WWW, ACM, 2019, pp. 2091–2102.

[45] N. Sachdeva, J. McAuley, How useful are reviews for recommendation? A critical review and potential improvements, in: SIGIR, ACM, 2020, pp. 1845–1848.

[46] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: ICLR, OpenReview.net, 2017.

[47] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: ICML, Proceedings of Machine Learning Research, vol. 70, PMLR, 2017, pp. 1263–1272.

[48] W.L. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: NIPS, 2017, pp. 1024–1034.

[49] J. Zhang, X. Shi, S. Zhao, I. King, STAR-GCN: stacked and reconstructed graph convolutional networks for recommender systems, in: IJCAI, ijcai.org, 2019, pp. 4264–4270.

[50] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, LightGCN: Simplifying and powering graph convolution network for recommendation, in: SIGIR, ACM, 2020, pp. 639–648.

[51] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, T. Tan, Session-based recommendation with graph neural networks, in: AAAI, AAAI Press, 2019, pp. 346–353.

[52] W. Fan, Y. Ma, Q. Li, Y. He, Y.E. Zhao, J. Tang, D. Yin, Graph neural networks for social recommendation, in: WWW 2019, ACM, 2019, pp. 417–426.

[53] M. Zhang, Y. Chen, Inductive matrix completion based on graph neural networks, in: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, OpenReview.net, 2020.

[54] F. Monti, M.M. Bronstein, X. Bresson, Geometric matrix completion with recurrent multi-graph neural networks, in: NIPS, 2017, pp. 3697–370.

[55] Y. Cen, X. Zou, J. Zhang, H. Yang, J. Zhou, J. Tang, Representation learning for attributed multiplex heterogeneous network, in: KDD, ACM, 2019, pp. 1358–1368.

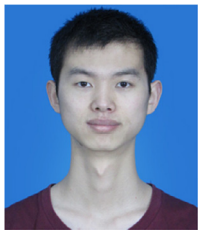
[56] S. Wang, L. Hu, Y. Wang, X. He, Q.Z. Sheng, M.A. Orgun, L. Cao, N. Wang, F. Ricci, P.S. Yu, Graph learning approaches to recommender systems: A review, CoRR (2020) abs/2004.11718.

[57] M. Jamali, M. Ester, A matrix factorization technique with trust propagation for recommendation in social networks, in: RecSys, ACM, 2010, pp. 135–142.

[58] G. Dror, N. Koenigstein, Y. Koren, M. Weimer, The Yahoo! music dataset and KDD-Cup '11, in: KDD, JMLR Proceedings. vol. 18, JMLR.org, 2012, pp. 8–18.

[59] B.N. Miller, I. Albert, S.K. Lam, J.A. Konstan, J. Riedl, MovieLens unplugged: experiences with an occasionally connected recommender system, in: Proceedings of the 8th International Conference on Intelligent User Interfaces, ACM, 2003, pp. 263–266.

[60] H. Guo, R. Tang, Y. Ye, Z. Li, X. He, DeepFm: A factorization-machine based neural network for CTR prediction, in: IJCAI, ijcai.org, 2017, pp. 1725–1731.



Hao Tang received the B.E. degree from The PLA Information Engineering University, Zhengzhou, China, in 2011, the M.E. degree from Shandong University of Science and Technology, QingDao, China, in 2013. After working for 5 years, he is currently working towards the Ph.D. degree at SMILES LAB, Xi'an Jiaotong University, Xi'an, China. His current research interests include recommender systems and social media big data mining.

Guoshuai Zhao received the B.E. degree from Heilongjiang University, Harbin, China, in 2012, the M.S. degree and Ph.D. degree from Xi'an Jiaotong University, Xi'an, China, in 2015 and 2019 respectively. He was an intern with the Social Computing Group at Microsoft Research Asia from January 2017 to July 2017, and was a visiting scholar with Northeastern University, U.S., from October 2017 to October 2018 and with MIT, U.S., from June 2019 to December 2019. Now he is an Associate Professor with Xi'an Jiaotong University. His research interests include social media big data analysis, recommender systems, and natural language generation.

Xuxiao Bu received the B.E. degree from Xi'an Jiaotong University, Xi'an, China, in 2017, and is expected to get the M.S. degree from Xi'an Jiaotong University in 2020. Her current research interests include social media big data analysis and recommender systems.

Xueming Qian received the B.S. and M.S. degrees from Xi'an University of Technology, Xi'an, China, in 1999 and 2004, respectively, and the Ph.D. degree from the School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an, China, in 2008. He was awarded the Microsoft Fellowship in 2006. From 1999 to 2001, he was an Assistant Engineer at Shanxi Daily. Since 2008, he has been an Associate Professor in the School of Electronics and Information Engineering, Xi'an Jiaotong University. Now, he is an Associate Professor in the School of Electronics and Information Engineering, Xi'an Jiaotong University. He is the Director of SMILES LAB. He was a Visiting Scholar at Microsoft Research Asia from August 2010 to March 2011. His research interests include social media big data mining and search. He is a member of the IEEE, ACM, and Senior Member of CCF.