Recurrent Neural Graph Collaborative Filtering

Beichuan Zhang, Zhijiao Xiao* and Shenghua Zhong
College of Computer Science and Software Engineering, Shenzhen University, China
zhangbeichuan2019@email.szu.edu.cn, {cindyxzj, csshzhong}@szu.edu.cn

***Abstract*—Collaborative Filtering (CF) is a prevalent technique in recommender systems. Substantial research focuses on learning the embedding of users and items via exploiting past user-item interactions. Recent years have witnessed the boom of Graph Convolutional Networks (GCNs) on CF. Performing graph convolution iteratively, GCN-based models concatenate/average/sum all outputs from different graph convolution layers to generate the embeddings of users and items. Although the previous methods have been proven effective, the pooling operations in the previous methods fail to consider the outputs from different graph convolution layers have different weights and the weights are related to sequential dependencies from precursor nodes. To resolve the aforementioned problems, in this work, we present a new model, Recurrent Neural Graph Collaborative Filtering (RNGCF), which proposes a sequential dependency construction module to adaptively generate the embeddings. Specifically, the module applies a gated recurrent unit (GRU) to learn the sequential dependencies from precursor nodes and an adaptive gated unit (AGU) to adaptively construct the embeddings based on the sequential dependencies. Extensive experiments on three benchmark datasets show that our model outperforms state-of-the-art models consistently. Our implementation is available in PyTorch [1].**

***Keywords*—Collaborative Filtering, Graph Convolutional Network, Recurrent Neural Network, Recommender System**

I. INTRODUCTION

To alleviate information overload on the web, recommender systems have been widely applied to many online services such as E-commerce and advertising [1], [2]. The goal of recommender systems is to predict whether a user will interact with an item, e.g., click, rate, purchase. As an effective solution, CF achieves the prediction via exploiting past user-item interactions. In CF, substantial research focuses on learning the embeddings of users and items to predict a user's preference for an item based on the similarity of the embeddings [2], [3]. As many user-item interaction data show graph structures, GCNs [4], [5] have been widely applied to CF [6], [7]. In GCNs, one graph convolution layer aggregates the features from nodes that are one hop away in the graph. This implies that the output of the k-th graph convolution layer aggregates the features from nodes that are k hops away in the graph [8]. Performing graph convolution iteratively, existing GCN-based works on CF concatenate/average/sum the all outputs from different graph convolution layers to generate the embeddings of users and items for combining all features from different hops [6], [7], [9].

[1]https://github.com/Book1996/RNGCF
DOI reference number: 10.18293/SEKE2021-015

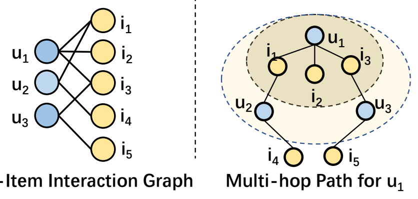

Fig. 1. An example of the paths rooted at u_1. Nodes in different color circles belong to different hops.

Although the previous methods have been proven effective, the pooling operations in the previous methods fail to consider the outputs from different graph convolution layers have different weights and the weights are related to the sequential dependencies from precursor nodes. In a user-item interaction graph, the explanations behind a progressive path unveil the nodes from different hops have different roles in reflecting the preference of users. For example, a bipartite user-item graph is denoted as Figure 1 left, where the edges between user u and item i are observed interactions. Examining the paths rooted at u_1 as Figure 1 right, as discuss in papers [6], [10], the first-hop path $u_1 \rightarrow \{i_1, i_2, i_3\}$ shows u_1's preference is related to the features of i_1, i_2 and i_3; the second-hop path $u_1 \rightarrow \{i_1, i_2, i_3\} \rightarrow \{u_2, u_3\}$ shows the behavioral similarity between u_1 and $\{u_2, u_3\}$, because u_2 and u_3 have interacted with the items that u_1 also has interacted with; the third-hop path $u_1 \rightarrow \{i_1, i_2, i_3\} \rightarrow \{u_2, u_3\} \rightarrow \{i_4, i_5\}$ shows u_1 may be interesting in i_4 and i_5, since u_1's similar users $\{u_2, u_3\}$ have consumed i_4 and i_5. This example shows the neighbor nodes from different hops have different roles in reflecting the preference of u_1. Furthermore, the roles of the nodes are related to the precursor nodes. In other words, the features of the nodes that are different hops away would have different weights to generate the embedding of u_1. Ignoring this fact, the existing works have several limitations: 1) Without adaptively constructing the embeddings could lead to the suboptimal representations of nodes; 2) Suppose there are two paths, the two paths are composed of the same nodes. But the orders of the nodes in the two paths are different. The average operations in the existing works may confuse the two paths to make the embeddings excessively similar.

To address the aforementioned problems, we propose a new GCN-based model, Recurrent Neural Graph Collaborative Filtering (RNGCF). RNGCF inputs a user's ID and a candidate item's ID and apply GCNs to output the user's preference for the candidate item. Different from GCN-based previous methods, we propose a sequential dependency construction module

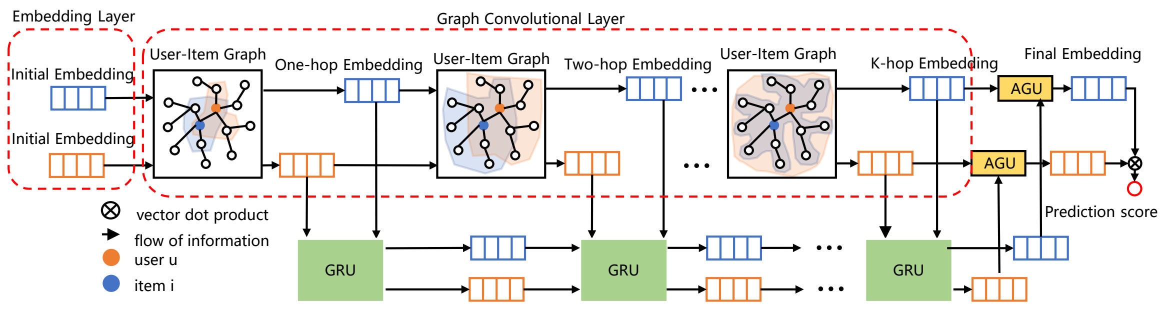

Fig. 2. An illustration of RNGCF model architecture. Arrowed lines present the flow of information. Orange presents user u, and blue presents item i.

to adaptive construct the embeddings of users and items. The construction module applies a GRU [11] to learn the sequential dependencies from precursor nodes and an adaptive gated unit (AGU) to adaptive construct the embeddings of users and items based on the sequential dependencies. We perform extensive experiments on three standard large real-world CF datasets, and the results clearly show our model can achieve better performance than state-of-the-art methods. To justify the designs in our model, we further conduct ablation studies on RNGCF. The results show each component of RNGCF has contributions to performance. To summarize, this work makes the following main contributions:

- We point out it is important to adaptively construct the embeddings of users and items based on the sequential dependencies from precursor nodes for our task.
- We propose a new GCN-based model RNGCF, which applies a gated recurrent unit (GRU) and an adaptive gated unit (AGU) to adaptively construct the embeddings of users and items based on the sequential dependencies.
- We demonstrate our model can achieve state-of-the-art results by extensive experiments on three standard large real-world CF datasets.

II. RELATED WORK

A. *Model-based CF Methods*

Collaborative Filtering (CF) is a prevalent technique in modern recommender systems. Among the various CF methods, item-based methods estimate a user's preference for an item via measuring the item's similarities with the items in her/his interaction history [12]. User-based methods estimate a user's preference for an item via finding similar users to the current user and then recommend the items in her/his similar users' interaction history. Other research focuses on learning the embedding of users and items by reconstructing historical user-item interactions. For example, Matrix Factorization (MF) [13] reconstructs historical user-item interactions via conducting inner product between the embeddings of users and items. BPR-MF [3] presented a pairwise ranking loss to optimize MF. NCF [14] pointed out that the inner product in MF had an inherent limitation and replaced the inner product with a multiple-layer perceptron (MLP). Another type of CF method does not project the IDs of each user as embedding vectors, the methods consider historical items of a given user as the embedding of the user [15], [16]. For example, SVD++ [15] regarded the weighted average of the embeddings of historical items as the embedding of users. With the development of attention mechanism, ACF [16] proposed to adaptive learn the weight of each historical item. We focus on how to learn the embedding parameters of users and items.

B. *Graph-based CF Methods*

Recent years have witnessed the boom of Graph Convolutional Networks (GCNs) [4], [5], [17]–[19]. As many real-world datasets in recommender systems show graph structures, researchers attempt to adopt GCNs for recommendation [1], [7], [20]. For example, Graph Convolutional Matrix Completion (GCMC) [21] proposed a graph auto-encoder framework to resolve matrix completion tasks. PinSage [1] combined random walk and graph convolution to handle recommendation tasks with billions of items and hundreds of millions of users. HOP-Rec [22] introduced confidence weighting parameters to incorporate graph convolution and random walk. NGCF [6] devised a new graph convolution layer to encode more collaborative signals into the embeddings of users and items. LightGCN [7] proved that the feature transformation and the nonlinear activation in NGCF [6] were useless in the recommendation task that only uses the IDs of users and items. NGAT4rec [9] employed a novel neighbor-aware graph attention layer that assigned different attention coefficients to the different neighbors of a given node. Although the previous methods have been proven effective, the pooling operations in the previous methods fail to consider the outputs from different graph convolution layers have different weights. Our GCN-based method applies a GRU and an AGU to adaptively construct the embeddings of users and items. There are other approaches that combine GCNs and recurrent architectures on different domains. For example, Evolving Graph Convolutional Networks (EGCN) [23] used a GRU to capture the dynamism of the graph sequence. For traffic prediction, Temporal Graph Convolutional Network (T-GCN) [24] adopted GCN to capture spatial dependence and GRU to capture temporal dependence.

III. METHOD

In this section, we introduce the architecture of our model in detail as shown in Figure 2, which includes an embedding layer, stacked graph convolution layers, a sequential dependency construction module.

A. Problem Formulation

In many real-world recommendation scenarios, user implicit data (e.g., click, rate, purchase) are more common than explicit data (e.g., ratings). Following previous GCNs based models [6], [7], we focus on implicit data. Suppose we have N users, M items and an user-item interaction graph $\mathcal{G}$. Based on $\mathcal{G}$, we can define $R \in \{0,1\}^{N\times M}$ as an implicit feedback interaction matrix. Entry R_{ui} in the interaction matrix R indicates whether user u interacted item i, which can be defined as follow:

$$R_{ui} = \begin{cases} 1 & \text{if (u,i) interaction is observed} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Typically, most of the entries in the interaction matrix R are unobserved (0). Based on the interaction matrix R, the aim of our task is to predict preference scores for unobserved entries in R.

B. Embedding Layer

Following most of recommender models [3], [25], [26], we create an embedding table $E^o \in \mathbb{R}^{(M+N)\times d}$ to project the IDs of N users and M items into initial vector representations, where d denotes the embedding size:

$$E^0 = [\underbrace{e^0_{u_1}, e^0_{u_2}, ..., e^0_{u_{N-1}}, e^0_{u_N}}_{user's\ embedding}, \underbrace{e^0_{i_1}, e^0_{i_2}, ..., e^0_{i_{M-1}}, e^0_{i_M}}_{item's\ embedding}]. \tag{2}$$

We represent the initial embeddings of user u and item i by $e^0_u \in \mathbb{R}^d$ and $e^0_i \in \mathbb{R}^d$, respectively. It is worth noting that these embeddings serve as the initial states for user u and item i, but are not embeddings that are used to predict. Thus, the superscript of the embeddings is 0.

C. Graph Convolutional Layer

Following the prior works [7], [10], we remove the two most common designs in GCNs: the feature transformation and the non-linear activate function. The stacked graph convolution layer in RNGCF is defined as:

$$\begin{aligned} e^k_u &= \frac{1}{|\mathcal{N}_u|+1} e^{(k-1)}_u + \sum_{j\in\mathcal{N}_u} \frac{1}{\sqrt{(|\mathcal{N}_u|+1)(|\mathcal{N}_j|+1)}} e^{(k-1)}_j, \\ e^k_i &= \frac{1}{|\mathcal{N}_i|+1} e^{(k-1)}_i + \sum_{j\in\mathcal{N}_i} \frac{1}{\sqrt{(|\mathcal{N}_i|+1)(|\mathcal{N}_j|+1)}} e^{(k-1)}_j, \end{aligned} \tag{3}$$

where $\mathcal{N}_u$ denotes the set of items that are interacted by user u, $\mathcal{N}_i$ denotes the set of users that interact with item i. $e^k_u \in \mathbb{R}^d$ and $e^k_i \in \mathbb{R}^d$ respectively represent the new embeddings of user u and item i after k graph convolutional layers.

D. Sequential Dependency Construction Module

1) Gated Recurrent Unit: After K graph convolutional layers, we can get two sequences $S_u = \{e^1_u, ..., e^K_u\}$, $S_i = \{e^1_i, ..., e^K_i\}$ for user u and item i, respectively. At present, the most widely used neural network models for processing sequence are the recurrent neural networks (RNNs) and Transformer [27]. Compared with LSTM [28] and Transformer [27], the GRU [11] has a relatively simple structure. Since LightGCN has proved that excessive parameters and nonlinear structures have no positive effect on the effectiveness of GCN based models on our task. Thus, we choose GRU to exploit the sequential dependencies in S_u and S_i:

$$\begin{aligned} c_u &= GRU(e^1_u, ..., e^K_u), \\ c_i &= GRU(e^1_i, ..., e^K_i), \end{aligned} \tag{4}$$

We regard $c_u \in \mathbb{R}^d$ and $c_i \in \mathbb{R}^d$ as the sequential dependencies in S_u and S_i, respectively. GRU: $\mathbb{R}^{K\times d} \to \mathbb{R}^d$

2) Adaptive Gated Unit: In order to fully consider the sequence dependence and the original graph convolution output, we propose an adaptive gated unit (AGU) to adaptive construct the final embeddings of users and items based on the sequential dependencies, which can be defined as:

$$\begin{aligned} e^f_u &= \sigma(Wc_u + b) \otimes e^K_u + e^K_u, \\ e^f_i &= \sigma(Wc_i + b) \otimes e^K_i + e^K_i. \end{aligned} \tag{5}$$

where $W \in \mathbb{R}^{d\times d}$, $b \in \mathbb{R}^d$ are learnable parameters. $\otimes$ is the element-wise product. σ is Sigmoid function.

E. Message and Node Dropout

Following the prior works [6], [21], we adopt two dropout techniques in RNGCF: message dropout and node dropout. Specifically, we apply the node dropout to randomly drop out some observed interactions. We apply the message dropout to drop out elements in Equation 4 and 5, which are updated as:

$$\begin{aligned} c_u &= GRU(Dropout(e^1_u, ..., e^K_u)), \\ c_i &= GRU(Dropout(e^1_i, ..., e^K_i)), \\ e^f_u &= Dropout(\sigma(Wc_u + b) \otimes e^K_u) + e^K_u, \\ e^f_i &= Dropout(\sigma(Wc_i + b) \otimes e^K_i) + e^K_i. \end{aligned} \tag{6}$$

F. Model Prediction

The model prediction is defined as the inner product of the final embeddings of user u and item i.

$$\hat{y}_{ui} = (e^f_i)^T e^f_u, \tag{7}$$

where $\hat{y}_{ui}$ is used as the ranking score for recommendation.

G. Model Training

As most of previous methods [7], [9], we employ Bayesian Personalized Ranking (BPR) loss [3] to optimize RNGCF. The objective function can be defined as:

$$L_{BPR} = -\sum_{u=1}^{N} \sum_{i\in N_u} \sum_{j\notin N_u} \ln \sigma\left(\hat{y}_{ui} - \hat{y}_{uj}\right) + \lambda||\mathbf{E^0}||^2, \tag{8}$$

where $\mathcal{N}_u$ denotes the set of items that user u interacted with. λ controls the L2 regularization strength to prevent overfitting. σ is Sigmoid function.

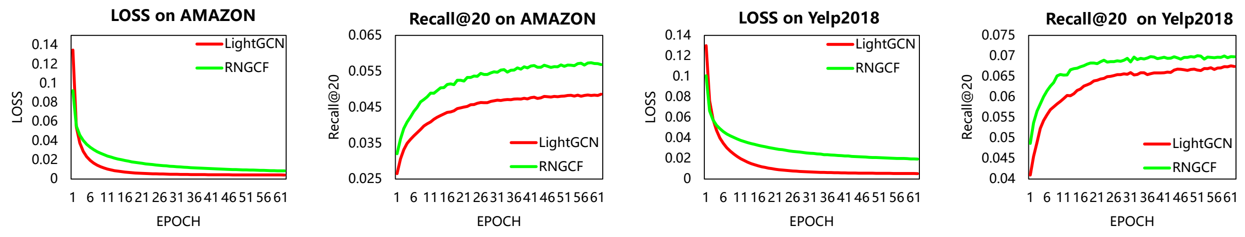

Fig. 3. Training curves of LightGCN and RNGCF, which are evaluated by training loss and testing recall@20 per 10 epochs on Yelp2018 and Amazon-Book.

Table I. Statistics of experimented data.

Dataset	User#	Item#	Interaction#	Density
Gowalla	29,858	40,981	1,027,370	0.00084
Yelp2018	31,668	38,048	1,561,406	0.00130
Amazon-Book	52,643	91,599	2,984,108	0.00062

IV. EXPERIMENTS

A. Experimental Settings

1) Baselines: NGCF [6] has shown to outperform several methods including GC-MC [21], PinSage [1], NeuMF [14], and HOP-Rec [22]. As the comparison is done on the same datasets under the same evaluation protocols, we do not further compare with these methods. To verify the effectiveness of our approach, we compare it with the following baselines:

- **MF** [3]: Matrix Factorization (MF) directly embedded user/item IDs as vectored representations and modeled user-item interaction with inner product.
- **NGCF** [6]: NGCF adopted three GCN layers on the user-item interaction graph to encode more neighbors' information into the embeddings of users and items.
- **LightGCN** [7]: LightGCN proved nonlinear transformation contributed little to the performance of NGCF.
- **DGCF** [10]: DGCF disentangled the representations of users and items at the granularity of user intents since a user generally had multiple intents to adopt certain items.
- **NIA-GCN** [2]: NIA-GCN proposed a cross-depth ensemble layer to preserve the relational information in neighborhood.
- **NGAT4rec** [9]: NGAT4rec generated the embeddings of neighbors according to the corresponding attention coefficients.

2) Dataset Description: To keep the comparison fair, we conduct experiments on three benchmark datasets: Gowalla, Yelp2018 and Amazon-Book. The three datasets are exactly as same as the LightGCN [7] and DGCF [10] paper used. For each dataset, 80% of historical interactions of each user are selected to constitute the training set, and the remaining historical interactions are treated as the test set. Yelp2018 is about local businesses like restaurants and bars. Amazon-Book is about the businesses like books. Gowalla contains user-venue check-in information from a location-based social network. The statistics of datasets are summarized in Table I.

3) Evaluation Metrics: To evaluate recommendation, we use the same protocols as previous methods [7], [10]: Recall@20 and NDCG@20. In the testing phase, we rank all items for a user and evaluate whether the score of the historical items is higher than all unobserved items'.

4) Parameter Settings: We implement our RNGCF model in PyTorch. We optimize RNGCF with Adam and use the default learning rate of 0.001. The default mini-batch size is 8192. We test the number of graph convolutional layers in the range of 1 to 4, and satisfactory performance can be achieved when the number of graph convolutional layers equals 3. The dropout rate is 0.3. Typically, 500 epochs are sufficient for RNGCF. For all methods, the embedding size d is searched in $\{16, 32, 64, 128\}$, in most cases, the optimal value is 128. The learning rate is searched in $\{0.005, 0.001, 0.0005, 0.0001\}$. The coefficient λ of L2 regularization term is tuned in $\{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$, in most cases the optimal value is 10^{-5}.

Table II. Overview performance comparison. Bold scores are the best and underlined scores are the second best.

Dataset	Gowalla		Yelp2018		Amazon	
Method	Recall	NDCG	Recall	NDCG	Recall	NDCG
MF	0.1388	0.1291	0.0533	0.0423	0.0350	0.0249
NGCF	0.1629	0.1355	0.0579	0.0477	0.0347	0.0281
NIA-GCN	0.1726	0.1358	0.0599	0.0491	0.0369	0.0287
LightGCN	0.1860	0.1554	0.0660	0.0521	0.0435	0.0328
DGCF	<u>0.1865</u>	<u>0.1562</u>	0.0670	0.0534	0.0440	0.0335
NGAT4rec	0.1855	0.1534	<u>0.0675</u>	<u>0.0554</u>	<u>0.0457</u>	<u>0.0358</u>
RNGCF	**0.1944**	**0.1628**	**0.0707**	**0.0582**	**0.0570**	**0.0442**
%Impro.	5.53%	4.22%	4.74%	5.05%	24.72%	23.46%

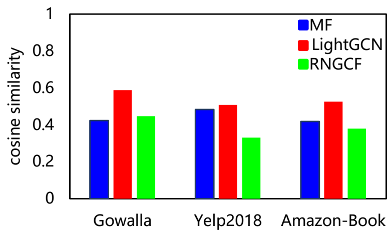

Fig. 4. Similarity of the embeddings.

B. Performance Comparison

Table II shows the best performance of all methods on three datasets. And Figure 3 shows the training curves of RNGCF and LightGCN, which are evaluated by training loss and testing Recall@20 per 10 epochs on Yelp2018 and Amazon-Book (results on Gowalla show similar trends which are omitted for space). Our method RNGCF achieves significant improvements over all methods across three datasets. In particular, RNGCF's relative improvements over the strongest baselines w.r.t. Recall@20 are **5.53%**, **4.76%**, and **24.72%** in Gowalla, Yelp2018, and Amazon-Book, respectively. This demonstrates the high effectiveness of our model. Across the three datasets, we find that the improvements on Amazon-Book are much more than that on the others. Compared with other datasets, Amazon-Book is the sparsest dataset. This suggests that RNGCF may be more suitable for sparse datasets.

Table III. Performance of RNGCF w.r.t different lengths of sequential dependencies. Bold scores are the best.

Dataset	Gowalla		Yelp2018		Amazon	
Length	Recall	NDCG	Recall	NDCG	Recall	NDCG
0	0.1850	0.1535	0.0670	0.0557	0.0501	0.0386
1	0.1913	0.1589	0.0688	0.0563	0.0555	0.0423
2	0.1923	0.1604	0.0695	0.0571	0.0562	0.0435
3(default)	**0.1944**	**0.1625**	**0.0707**	**0.0582**	**0.0575**	**0.0442**

Table IV. Performance of RNGCF w.r.t different orders of sequential dependencies. Bold scores are the best.

Dataset	Gowalla		Yelp2018		Amazon	
Order	Recall	NDCG	Recall	NDCG	Recall	NDCG
Default	**0.1944**	**0.1625**	**0.0707**	**0.0582**	**0.0575**	**0.0442**
Reverse	0.1900	0.1589	0.0683	0.0551	0.0513	0.0395
Rand-1	0.1913	0.1597	0.0681	0.0533	0.0501	0.0388
Rand-2	0.1918	0.1602	0.0682	0.0553	0.0515	0.0397

C. Impact of Sequential Dependencies

1) Analysis of Embedding Similarity: As we analyze in Section I, the average operation in LightGCN [7] may confuse the orders of high-order paths, which may make the embeddings of nodes excessive similar. To verify this, we define the average cosine similarity of the embeddings of interconnected nodes:

$$\frac{1}{|R^{+}|}\sum_{(u,i)\in\mathcal{R}+}\left(\frac{e_u e_i^T}{||e_u||\times||e_i||}\right), \tag{9}$$

where R^{+} is a set of the observed interactions. Figure 4 shows the similarity of embeddings learned by three models (MF, LightGCN, and RNGCF). The similarity of embeddings learned by LightGCN is higher than that of MF. This indicates graph convolution makes the embeddings more similar, which is consistent with LightGCN's finding. The similarity of embeddings learned by RNGCF is lower than that of LightGCN, but the performance of RNGCF does not decrease. This demonstrates that the average operation may make embeddings excessive similar.

2) Impact of Sequential Dependencies on Prediction: To discuss the impacts of sequential dependencies on prediction, we feed different variants of the outputs from different graph convolution layers into GRU. Table III and Table IV show the impacts of the length and the order of the outputs. Note that we sample two kinds of outputs in random order to alleviate the impacts of accidental factors. As the length of the outputs increases from 0 to 3, the performance on the three datasets increases. This indicates **complete** sequential dependencies are important to prediction. Shuffling the outputs achieves worse results. This indicates **correct** sequential dependencies can lead to good performance.

D. Ablation Studies

We perform ablation studies on RNGCF to show how the components of RNGCF affect performance. Table V shows the results of RNGCF and its variants on three datasets.

- **Remove Dropout**: We conduct RNGCF without Dropout on three datasets. We find that the performance on three datasets is significantly worse. The results show that the dropout can effectively regularize our model to achieve better performance.
- **Remove GRU**: RNGCF without GRU achieves poor results on three datasets. GRU is used to exploit the sequential dependencies in multi-hop paths. The results may indicate the importance of the sequential dependencies.

Table V. Performance of RNGCF and its variants. Bold scores are the best and underlined scores are the second best.

Dataset	Gowalla		Yelp2018		Amazon-Book	
Method	Recall	NDCG	Recall	NDCG	Recall	NDCG
RNGCF	**0.1944**	**0.1628**	**0.0707**	**0.0582**	**0.0570**	**0.0442**
Remove GRU	0.1850	0.1535	0.0688	0.0565	0.0501	0.0386
Remove Dropout	0.1841	0.1507	0.0673	0.0544	0.0541	0.0426
Remove AGU	0.1563	0.1252	0.0526	0.0433	0.0409	0.0323
Transformer	0.1894	0.1580	0.0690	0.0564	0.0546	0.0423
LSTM	0.1874	0.1562	0.0689	0.0563	0.0487	0.0377

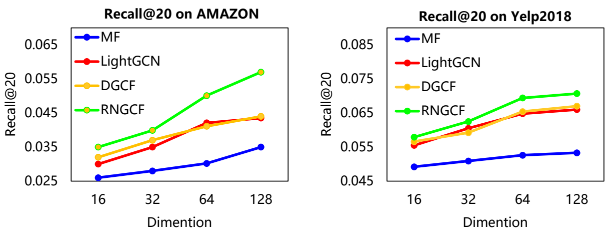

Fig. 5. Performance of NGAT4rec, LightGCN and RNGCF w.r.t different dimensions on Yelp and Amazon-Book. w.r.t different dimension.

- **Remove AGU**: We propose a AGU to adaptively construct the embeddings of users and items based on the sequential dependencies. Without AGU, RNGCF achieves poor results on all datasets. This demonstrates that directly taking the output of GRU as the final embeddings can not get better results.
- **Other Common Aggregation Functions**: We conduct RNGCF with other common aggregation functions instead of GRU on three datasets to demonstrate the effectiveness of GRU, such as Transformer [27] and LSTM [28]. Other common aggregation functions achieve worse performance on three datasets than GRU. This demonstrates the effectiveness of GRU. Compared with LSTM and Transformer, the GRU has a relatively simple structure but gets better performance. This may indicate excessive parameters in model are not suitable for our tasks.

E. Hyper-parameter Studies

1) Study on dimension of embeddings: We conduct a dimension study on MF, LightGCN, RNGCF, NGAT4rec on Yelp2018, and Amazon-Book. The results of the experiments are shown in Figure 5. As the dimension increases from 16 to 128, the performance of all models increases. The five methods all apply dot product to compute the relevance of items and users. But the limitations of dot product function are well documented in the literature [14]. Thus the results may indicate that the limitations of dot product function will become weaker as the dimension increases. RNGCF outperforms all models on all dimensions, which more forcefully indicates RNGCF is effective.

2) Study on number of feature aggregation layers: Table VI shows the performance at different layers (from 1 to 3) and the percentage of relative improvement on each metric. As the number of feature aggregation layers increases from 1 to 3, the performance of all models increases. RNGCF outperforms all models on all dimensions. In particular, RNGCF's relative improvements over the strongest baselines w.r.t. Recall@20 are the largest when the number of layers reaches 3.

V. CONCLUSION

In this work, we propose a new framework named RNGCF. RNGCF takes a user's ID and a candidate item's ID as inputs and apply GCNs to output the user's preference for the candidate item.

Table VI. Performance comparison of LightGCN, DGCF,NGAT and our model w.r.t. number of layers.

Datasets		Gowalla		Yelp2018		Amazon-Book	
Layer#	Method	Recall	NDCG	Recall	NDCG	Recall	NDCG
1 Layer	LightGCN	0.1755	0.1315	0.0631	0.0515	0.0384	0.0298
	DGCF	0.1794	0.1521	0.0640	0.0522	0.0399	0.0308
	NGAT4rec	0.1715	0.1298	0.0613	0.0504	0.0347	0.0281
	RNGCF	**0.1849**	**0.1554**	**0.0660**	**0.0532**	**0.0445**	**0.0337**
	Improv.	(3.11%)	2.21%	3.13%	(2.02%)	(11.55%)	(9.73%)
2 Layers	LightGCN	0.1777	0.1524	0.0622	0.0504	0.0411	0.0315
	DGCF	0.1834	0.1560	0.0653	0.0532	0.0422	0.0324
	NGAT4rec	0.1757	0.1514	0.0656	0.0540	0.0434	0.0339
	RNGCF	**0.1887**	**0.1593**	**0.0675**	**0.0551**	**0.0483**	**0.0371**
	Improv.	(2.90%)	(2.16%)	(3.23%)	(2.12%)	(11.38%)	(9.52%)
3 Layers	LightGCN	0.1860	0.1554	0.0660	0.0521	0.0435	0.0328
	DGCF	0.1865	0.1562	0.0670	0.0534	0.0440	0.0335
	NGAT4rec	0.1855	0.1534	0.0675	0.0554	0.0457	0.0358
	RNGCF	**0.1944**	**0.1628**	**0.0707**	**0.0582**	**0.0570**	**0.0442**
	Improv.	5.53%	4.22%	4.74%	5.05%	24.72%	23.46%

Different from GCN-based previous methods on our task, we design an adaptive unit to adaptively construct the embeddings of users and items based on the sequential dependencies. Extensive experiments on three benchmark datasets show that our model outperforms state-of-the-art models consistently and each component of RNGCF is effective. In particular, RNGCF's relative improvements over the strongest baselines w.r.t. Recall@20 are **5.53%**, **4.76%**, and **24.72%** in Gowalla, Yelp2018, and Amazon-Book, respectively. In future work, we will study how to exploit auxiliary information such as item knowledge graphs, social networks, and multimedia content for our task.

Acknowledgment

This work was supported by the National Natural Science Foundation of China [No. 62002230], the Natural Science Foundation of Guangdong Province [No. 2019A1515011181], the Science and Technology Innovation Commission of Shenzhen under Grant [No. JCYJ20190808162613130], and the Shenzhen high-level talents program.

References

[1] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 974–983.

[2] J. Sun, Y. Zhang, W. Guo, H. Guo, R. Tang, X. He, C. Ma, and M. Coates, "Neighbor interaction aware graph convolution networks for recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1289–1298.

[3] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," *arXiv preprint arXiv:1205.2618*, 2012.

[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[5] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in neural information processing systems*, 2017, pp. 1024–1034.

[6] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 2019, pp. 165–174.

[7] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," *arXiv preprint arXiv:2002.02126*, 2020.

[8] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International conference on machine learning*. PMLR, 2019, pp. 6861–6871.

[9] J. Song, C. Chang, F. Sun, X. Song, and P. Jiang, "Ngat4rec: Neighbor-aware graph attention network for recommendation," *arXiv preprint arXiv:2010.12256*, 2020.

[10] X. Wang, H. Jin, A. Zhang, X. He, T. Xu, and T.-S. Chua, "Disentangled graph collaborative filtering," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1001–1010.

[11] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[12] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 285–295.

[13] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[14] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[15] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 426–434.

[16] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua, "Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2017, pp. 335–344.

[17] J. Wu, S.-h. Zhong, and Y. Liu, "Dynamic graph convolutional network for multi-video summarization," *Pattern Recognition*, vol. 107, p. 107382, 2020.

[18] Z. Zhang, M. Zheng, S. Zhong, and Y. Liu, "Steganographer detection via a similarity accumulation graph convolutional network," *Neural Networks*, vol. 136, pp. 97–111, 2021.

[19] Y. Li, C. Yin, and S. Zhong, "Sentence constituent-aware aspect-category sentiment analysis with graph attention networks," in *Natural Language Processing and Chinese Computing - 9th CCF International Conference, NLPCC 2020, Zhengzhou, China, October 14-18, 2020, Proceedings, Part I*, vol. 12430. Springer, 2020, pp. 815–827.

[20] W. Fang and L. Lu, "Deep graph attention neural network for click-through rate prediction," in *The 32nd International Conference on Software Engineering and Knowledge Engineering, SEKE 2020, KSIR Virtual Conference Center, USA, July 9-19, 2020*, 2020, pp. 483–488.

[21] R. van den Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *CoRR*, 2017.

[22] J.-H. Yang, C.-M. Chen, C.-J. Wang, and M.-F. Tsai, "Hop-rec: high-order proximity for implicit recommendation," in *Proceedings of the 12th ACM Conference on Recommender Systems*, 2018, pp. 140–144.

[23] A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, H. Kanezashi, T. Kaler, T. Schardl, and C. Leiserson, "Evolvegcn: Evolving graph convolutional networks for dynamic graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 5363–5370.

[24] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-gcn: A temporal graph convolutional network for traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3848–3858, 2019.

[25] N. Mu, D. Zha, L. Zhao, and R. Gong, "Collaborative denoising graph attention autoencoders for social recommendation," in *The 32nd International Conference on Software Engineering and Knowledge Engineering, SEKE 2020, KSIR Virtual Conference Center, USA, July 9-19, 2020*, 2020, pp. 519–524.

[26] Y. Wang, K. Shi, and Z. Niu, "A session-based job recommendation system combining area knowledge and interest graph neural networks," in *The 32nd International Conference on Software Engineering and Knowledge Engineering, SEKE 2020, KSIR Virtual Conference Center, USA, July 9-19, 2020*, 2020, pp. 489–492.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.