



# Refined Node Type Graph Convolutional Network for Recommendation

Wei He<sup>1</sup>, Guohao Sun<sup>1</sup>, Jinhua Lu<sup>1</sup>, Xiu Fang<sup>1(✉)</sup>, Guanfeng Liu<sup>2</sup>,  
and Jian Yang<sup>2</sup>

<sup>1</sup> Donghua University, Shanghai, China  
xiu.fang@dhu.edu.cn

<sup>2</sup> Macquarie University, Sydney, Australia

**Abstract.** Recently, because of the remarkable performance in alleviating the data sparseness problem in recommender systems, Graph Convolutional (Neural) Networks (GCNs) have drawn wide attention as an effective recommendation approach. By modeling the user-item interaction graph, GCN iteratively aggregates neighboring nodes into embeddings of different depths according to the importance of each node. However, the existing GCN-based methods face the common issues that, they do not consider the node information and graph structure during aggregating nodes, such that they cannot assign reasonable weights to the neighboring nodes. Additionally, they ignore the differences in node types in the user-item interaction graph and thus, cannot explore the complex relationship between users and items, resulting in a suboptimal result. To solve these problems, a novel GCN-based framework called **RNT-GCN** is proposed in this paper. RNT-GCN integrates the structure of the graph and node information to assign reasonable importance to different nodes. In addition, RNT-GCN refines the node types, such that the heterogeneous properties of the user-item interaction graph can be better preserved, and the collaborative information of users and items can be effectively extracted. Extensive experiments prove the **RNT-GCN** achieved significant performance compared to **SOTA** methods.

**Keywords:** Recommender system · Graph convolutional network · User-item interaction graph

## 1 Introduction

Personalized recommender systems play a vital role in the era of big data and can extract pivotal information from a mass of data and effectively alleviate information overload [4]. Recommender systems face the common issue that the data in the real world is extremely sparse and noisy [7, 22], in other words, numerous items interact with only a few users. Therefore, it is hard to extract

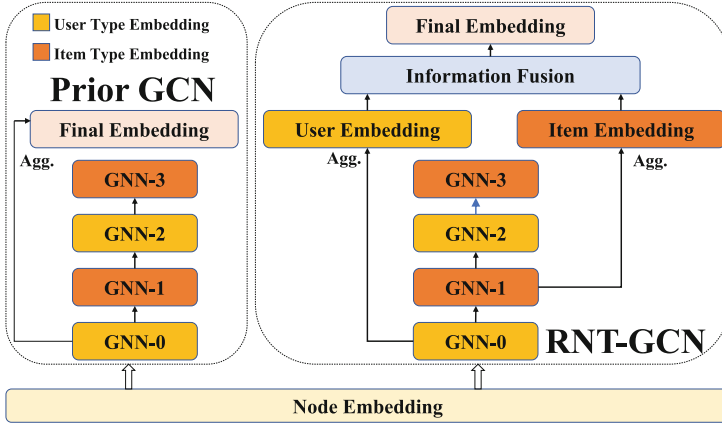
---

W. He and G. Sun—Contributed equally to this research.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023  
X. Yang et al. (Eds.): ADMA 2023, LNAI 14176, pp. 91–106, 2023.  
[https://doi.org/10.1007/978-3-031-46661-8\\_7](https://doi.org/10.1007/978-3-031-46661-8_7)

enough information to predict user preferences accurately, and the performance of recommendations is limited in real-world applications.

Recently, Graph Convolutional (Neural) Networks (GCNs) with their powerful representation learning capabilities [11] has attracted wide attention in many areas, such as Natural Language Processing (NLP) [15], etc. By modeling the historical interaction data as a graph (e.g., the user-item interaction graph), GCN-based methods successfully extracted the high-order collaboration signals between nodes and effectively alleviated the data sparsity problem [7, 13, 22].



**Fig. 1.** The difference between the prior GCN-based methods and RNT-GCN.

Although remarkable progress has been made with GCN-based methods, there are still some significant disadvantages. When aggregating the information of neighbor nodes, GCN-based methods cannot assign reasonable importance to different neighbor nodes because they only consider the importance of neighbor nodes based on the structure of the graph or node information. To be specific, the methods of using the structure of the graph aggregate neighbor node according to the degree of the node, the popular nodes (with larger degrees) have a higher possibility to be recommended. The methods of using the node information aggregate neighbor node according to the embedded similarity between the central node and the neighbor node, the neighbor nodes with low similarity probably contain important information and will be assigned a low weight. These disadvantages will result in the neglect of important information and thus, obtaining poor recommendations. In addition, the existing methods directly process the user-item interaction graph and do not consider the intrinsic differences between user and item nodes, resulting in low-quality embeddings. For example, as shown in Fig. 1, the prior GCN-based methods do not distinguish the user and item nodes, they indiscriminately aggregate embeddings at different depths. Therefore, due to the lack of distinguishing the node type, the existing methods cannot explore the complex relationship between users and items, which results

in a suboptimal result. These disadvantages limit the expression power of GCN, which influences the recommendation performance of the GCN-based methods.

To solve the above-mentioned problems, we propose a novel **Refined Node Type Graph Convolutional Network for Recommendation**, called **RNT-GCN**. The main contributions of this paper can be summarized as follows:

- In RNT-GCN, we propose an attention-refinement network (ARN) to assign the appropriate aggregation importance for different neighbor nodes. ARN learns the relationship between neighbor nodes and central nodes at the graph level and node level, such that the important information of local neighbor nodes can be reasonably extracted.
- In RNT-GCN, we propose a type-aware embedding aggregation strategy. By this strategy, the nodes of each layer are aggregated according to their type, such that the heterogeneous properties of the user-item interaction graph can be better preserved.
- In RNT-GCN, we propose a new information fusion layer that captures user and item connections during prediction. This layer considers the connections between different attribute nodes on different sides during prediction and effectively extracts the collaborative information of users and items.
- Extensive experiments on four public datasets demonstrate that RNT-GCN outperforms the SOTA models. The code is published on GitHub<sup>1</sup>.

## 2 Related Work

Some scholars [3, 7, 13, 14, 17, 20, 22] realized that the historical interaction between users and items can be modeled as a user-item interaction graph. It can provide plentiful information about the graph structure and nodes and effectively improve the performance of the recommender system. Early works related to graphs mainly explored the information of the graph through random walks such as randwalk [6] etc. However, the performance of these methods is heavily influenced by the quality of interactions generated by the randomwalk [13]. As a consequence, models become difficult to train.

In recent years, many GCN-based methods have been proposed to explore graph-structured data. For example, by performing embedding propagating on the graph, NGCF [22] explicitly explores the higher-order connections between nodes. To exploit the heterogeneous nature of the user-item bipartite graph, NIA-GCN [17] parallel independently learns the representation of the user and item from different neighborhood depths and calculates the inner product between the first and second layer embeddings, resulting in only the node information of the specific range of receptivity fields can be captured. These GCN-based methods face many issues, such as complexity, bulkiness, and over-smoothing [2, 13], which result in the shallow depth of the convolutional layer and then make these methods hard to extract the information of the high-order node. To solve these problems, some works try to simplify GCN [3, 7, 13, 14]. For example, in order to

<sup>1</sup> <https://github.com/WeiHeCnSH/RNTGCN-TF-master>.

better learn the attributes of neighbors, JK-NET [23] aggregates the embedding of each layer in the jump concatenate and adaptively selects the range of aggregated neighborhood information. Similar to JK-Net, LR-GCCF [3] uses residuals to concatenate the embeddings of the layers and removes nonlinearities. Meanwhile, LightGCN [7] realized that nodes encoded with ID contain only a small amount of semantic information, and feature transformation is also redundant for GCN. Hence, GCN is further simplified. Later, based on LightGCN, IMP-GCN [13] and UltraGCN [14] are proposed. Compared with LightGCN, IMP-GCN [13] thinks that the indiscriminate use of higher-order neighbor nodes allows harmful information to be propagated in the graph. To solve this problem, they propose to divide the subgraph according to the user’s interest and perform high-order convolution operations in the subgraph. UltraGCN [14] thinks that the messaging mechanism in GCN greatly slows down the convergence of the model, so they further simplify LightGCN by directly approximating the limit of infinite-layer graph convolutions via a constraint loss. In addition, GDCF [24] think existing approaches model interactions within the euclidean space be inadequate to fully uncover the intent factors between user and item. Meanwhile, SGNR [18] separates the user-item interaction graph into two weighted homogeneous graphs to resolve the heterogeneous information aggregation problem, but doesn’t establish the distinction and relationship between different types of embeddings of the users and items.

Although these graph-based approaches have achieved good performance, they still face the problems proposed in the Introduction section.

### 3 Problem Statement

**User-item Interaction Matrix:** An user-item interaction matrix is denoted as  $A \in \mathbb{R}^{M \times N}$ , where  $M$  indicates the number of users and  $N$  indicates the number of items. If there is an interaction between user  $u$  and item  $i$ , then  $a_{u,i} = 1$ , otherwise  $a_{u,i}$  will be 0.

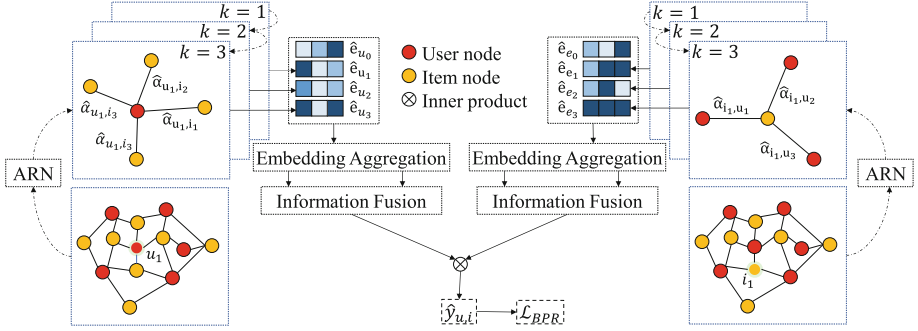
**Bipartite Graph:** A bipartite graph  $\mathcal{G}(\mathcal{U}, \mathcal{I}, \mathcal{E})$  is constructed based on the user-item interaction matrix, where  $\mathcal{U}$  indicates user set,  $\mathcal{I}$  indicates item set, and  $\mathcal{E}$  indicates edge set if user  $u$  interacte with item  $i$ . Each node in the graph can be represented using a vector of dimension  $d$ .  $e_u^{(0)} \in \mathbb{R}^d$  and  $e_i^{(0)} \in \mathbb{R}^d$  represents the initialized embedding of the user  $u$  and the item  $i$ , respectively.

**Target Problem:** Our goal is to train a model to predict whether an interaction will occur between the user  $u$  and the item  $i$  in the bipartite graph  $\mathcal{G}(\mathcal{U}, \mathcal{I}, \mathcal{E})$ .

### 4 Methodology

In this section, we introduce RNT-GCN in detail. As shown in Fig. 2. The proposed RNT-GCN consists of three parts: the attention refinement network, the type-aware embedding aggregation, and the information fusion layer. The user

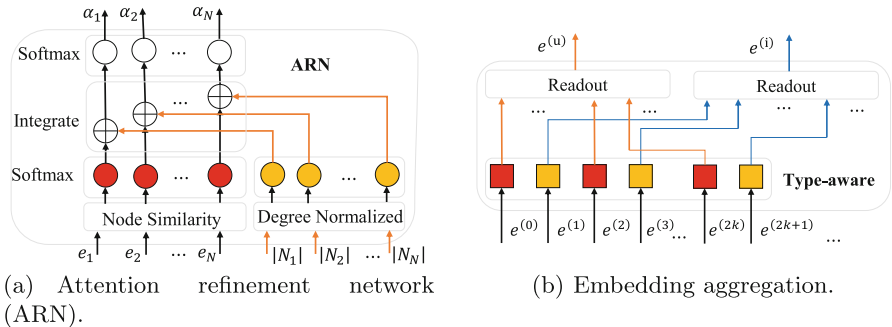
(left) and the item (right) first perform multi-layer convolution to compute the representation of each layer, then aggregate the information of each layer according to the node type, and finally fuse the embedding information of both node types to form the final node representation to make predictions.



**Fig. 2.** The overall framework of the proposed RNT-GCN model.

#### 4.1 Attention Refinement Network

To retain important information and weaken the influence of unimportant nodes, by using the attention refinement network (ARN) in Fig. 3.a, we combine the structure of the graph and the information of the nodes to assign reasonable aggregate importance to neighboring node from the graph level and the node level, respectively.



**Fig. 3.** Illustration of proposed attention refinement network (left) and embeddings aggregation (right).

**Importance of Graph-Level.** Inspired by the famous GCN-based methods [3, 7, 22], we use symmetric normalization terms to express the connection strength between different neighbors  $i$  and the center  $u$  at the graph level. In this way, we combine the information of the degree of the central node and the neighboring node. The importance of the graph level can be represented as follows:

$$\hat{\alpha}_{u,i}^{(g)} = \frac{|N_u \cap N_i|}{\sqrt{|N_u|}\sqrt{|N_i|}} = \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}}. \quad (1)$$

where  $N_u$  and  $N_i$  indicate the number of the first-order neighbors of users and items, respectively. The intersection between the neighbors of users and items can be represented as the interaction generated by users and items.

**Importance of Node-Level.** To better aggregate node information, we further consider the relationship of nodes represented in the embedding space to assign importance to different neighbor nodes. To be specific, in the embedding space, the distance between two nodes can reflect the correlations between the nodes to some extent. The nodes with closer distances indicate more similarities, so more information should be aggregated from these nodes [19, 21]. We assign the importance of node level according to the distance between nodes. The importance of node level can be represented as follows:

$$\alpha_{u,i}^{(n)} = \text{sim}_n \left( e_u^{(0)}, e_i^{(0)} \right). \quad (2)$$

For simplicity, we use cosine similarity to measure the strength of the connection between nodes. In the future, we will research more node-level importance calculation methods. The importance of node level is represented as follows:

$$\alpha_{u,i}^{(n)} = \cos \left( e_u^{(0)}, e_i^{(0)} \right) = \frac{|e_u^{(0)} \odot e_i^{(0)}|}{|e_u^{(0)}||e_i^{(0)}|}. \quad (3)$$

where  $\odot$  refers to the element-wise product. We normalize all node cosine similarities as node-level importance using the softmax function. The normalized node level importance can be represented as follows:

$$\hat{\alpha}_{u,i}^{(n)} = \text{softmax} \left( \alpha_{u,i}^{(n)} \right) = \frac{\exp \left( \alpha_{u,i}^{(n)} \right)}{\sum_{j \in N_u} \exp \left( \alpha_{u,j}^{(n)} \right)}. \quad (4)$$

The node-level importance of the item can also be obtained in the same way.

**Importance of Integration.** To integrate the importance of two different neighbor nodes, we designed a dual attention integration strategy to combine the structure of the graph and the relationship between nodes as the importance of different neighboring nodes. It can be represented as follows:

$$\alpha_{u,i} = \lambda \hat{\alpha}_{u,i}^{(n)} + (1 - \lambda) \alpha_{u,i}^{(g)}. \quad (5)$$

where  $\lambda$  is an adjustable super-parameter for balancing the weight of the importance of the two different aspects. The importance of different neighbor nodes during aggregated can be represented as follows:

$$\hat{\alpha}_{u,i} = \text{softmax}(\alpha_{u,i}/\tau) = \frac{\exp(\alpha_{u,i}/\tau)}{\sum_{j \in N_u} \exp(\alpha_{u,j}/\tau)}. \quad (6)$$

where  $\tau$  indicates the temperature of the normalization function, which can make the training process smoother. The normalized importance of nodes can avoid the scale increase of embedding in the convolution.

Same as [7], we also abandon feature transformation and nonlinear activation. Meanwhile, we have also removed the self-loop, such that there is only one type of neighbor node per layer. Therefore, the graph convolution operation in RNT-GCN can be represented as follows:

$$e_u^{(l+1)} = \sum_{i \in N_u} \hat{\alpha}_{u,i} e_i^{(l)}, e_i^{(l+1)} = \sum_{u \in N_i} \hat{\alpha}_{u,i} e_u^{(l)}. \quad (7)$$

where  $e_u^{(l)}$  and  $e_i^{(l)}$  denote the embedding representation of user node  $u$  and item node  $i$  at  $l$  layer, respectively.

## 4.2 Type-Aware Embedding Aggregation

To capture neighbor node embedding information at different depths, we perform  $L$  convolution operations to obtain the embedding of the user (item). Because of the heterogeneous property of the user-item interaction graph, the embedding of different layers contains different semantic information. As shown in Fig. 1, the embedding of user types is obtained from the even layer (i.e., 0, 2,...), and item type is obtained from the odd layer (i.e., 1, 3,...). On the item side, similar findings can be derived by analyzing the convolution process of the items. To preserve the embedding information of different types of nodes, we designed a node-type-aware information aggregation strategy, as shown in Fig. 3.b. Based on the node types in each layer, we divide the embedding of the convolution layers into two types, i.e., the embedding of user type  $e^{(u)}$  and embedding of item type  $e^{(i)}$ . There, we aggregate embeddings of different depths according to the node type to generate embeddings of different types of nodes. On the user side, the odd layer is the embedding of item types  $e_u^{(i)}$ , and the even layer is the embedding of user types  $e_u^{(u)}$ . The user and item type embeddings for user nodes can be represented as follows:

$$e_u^{(u)} = \sum_{k=0,1,2,\dots} a_k e_u^{(2k)}, e_u^{(i)} = \sum_{k=0,1,2,\dots} b_k e_u^{(2k+1)}. \quad (8)$$

On the item side, the odd layer is the embedding of user types  $e_i^{(u)}$ , and the even layer is the embedding of item types  $e_i^{(i)}$ . Similarly, the user type embeddings and item type embeddings for item nodes can be represented as follows:

$$e_i^{(i)} = \sum_{k=0,1,2,\dots} c_k e_i^{(2k)}, e_i^{(u)} = \sum_{k=0,1,2,\dots} d_k e_i^{(2k+1)}. \quad (9)$$

where  $a_k, b_k, c_k, d_k$  indicate the aggregation weights of  $k$  layer embeddings of the same type of nodes on the user and item sides, respectively. Here, we average the embedding of each layer.

### 4.3 Information Fusion

The existing embedding aggregation methods [1, 3, 7, 22] do not distinguish node types. Therefore, the user-item relationship will appear in the prediction process, which will affect the recommendation performance. To solve this problem, we design two novel information fusion methods, i.e., sequence concatenate and reverse sequence concatenate, considering two different convolutional processes, to explore the impact of different combinatorial relationships of embedding types on the results.

**Sequence Concatenate.** The embedding of two node types is concatenated according to the order of embedding generated by the convolution process to generate the final representation of nodes. The final representation of the node can be represented as follows:

$$e_u = e_u^{(u)} \| e_u^{(i)}, e_i = e_i^{(i)} \| e_i^{(u)}. \quad (10)$$

where  $\|$  denotes the concatenate operation. The informational fusion strategy of sequence concatenate is simply denoted as RNT-sc.

**Reverse Sequence Concatenate.** Consequently, we adjust the order of concatenating embeddings so that the same types of embedding correspondence are used during prediction. The final representation of the node can be represented as follows:

$$e_u = e_u^{(u)} \| e_u^{(i)}, e_i = e_i^{(u)} \| e_i^{(i)}. \quad (11)$$

The informational fusion strategy of reverse-sequence concatenate is simply denoted as RNT-rsc.

### 4.4 Model Prediction

Similar to the existing GCN-based methods [1, 3, 7, 13, 17, 22], we use the inner product of the final representation of the user and the item as the preference prediction of the user  $u$  for the item  $i$ . It can be represented as follows:

$$\hat{r}_{u,i} = e_u^T e_i. \quad (12)$$

We expand the scoring function to simply investigate the impact of two different information-fusion strategies on the results. The scoring prediction for sequential concatenate expands can be represented as follows:

$$\hat{r}_{u,i} = e_u^{(u)} \odot e_i^{(i)} + e_u^{(u)} \odot e_i^{(i)}. \quad (13)$$



The prediction score for the reverse concatenate expands can be represented as follows:

$$\hat{r}_{u,i} = e_u^{(u)} \odot e_i^{(u)} + e_u^{(i)} \odot e_i^{(i)}. \quad (14)$$

Sequential concatenate uses embeddings of user types and item types to perform inner product operations and force them to be close to each other in different semantic spaces. Inverse concatenate performs inner product operations in homogeneous embeddings, such that the intrinsic properties of different types of embeddings are preserved.

## 4.5 Model Training

The Bayesian Personalized Ranking (BPR) [16] loss is applied as the loss function of RNT-GCN. The loss function can be represented as follows:

$$\mathcal{L}_{BPR} = \frac{1}{|\mathcal{O}|} \sum_{(u,i,j) \in \mathcal{O}} \ln \sigma(\hat{r}_{u,i} - \hat{r}_{u,j}) + \gamma \|\Theta\|_2^2. \quad (15)$$

where  $\mathcal{O}$  indicates the training set,  $i$  indicates the observed positive samples, and  $j$  is a negative sample randomly sampled from items with which the user has not interacted. The  $\gamma$  indicates the  $L_2$  regularization weights and the  $\Theta$  indicates the embedding of the user with the item nodes and the trainable parameters in the model. The  $\sigma$  indicates the Sigmoid function, and the Adam algorithm [10] is used to optimize the model parameters.

# 5 Experimental

## 5.1 Experimental Setup

In this section, we conduct extensive experiments to compare the performances of our proposed RNT-GCN models. The experiments will answer the following several questions:

- Q1: How is the performance of RNT-GCN compared with the SOTA graph-based methods?
- Q2: What is the impact of different information fusion strategies on the performance of RNT-GCN?
- Q3: What is the performance impact of interaction groups with different sparsities levels?
- Q4: How effect of different parameters (number of layers  $L$ , Balance Weight  $W$ , Node Dropout Ratio  $P$ ) on RNT-GCN performance?
- Q5: How do the ARN, type-aware embedding aggregation, and the information fusion layer contribute to the performance of RNT-GCN?
- Q6: Whether RNT-GCN can improve the embedding quality of nodes?

*DataSet.* In order to objectively evaluate our proposed method, we conducted experiments on four publicly available datasets, **Kindle Store** (14,356 users, 15,885 items, 357,477 interactions), **Gowalla** (29,858 users, 40,981 items, 1,027,370 interactions), **ml-1m** (6,022 users, 3,043 items, 895,699 interactions), and **Book-Cross** (6,768, 13,683 items, 374,563 interactions)<sup>2</sup>. The preprocessed datasets of the first three are provided by **IMP-GCN** [13], **LightGCN** [7], and **UltraGCN** [14], respectively. For the Book-Cross dataset, we treat all ratings as interaction records and remove the users and items with fewer than 10 interaction records. For each dataset, 80 percent of each user’s interaction records is randomly selected as the training set, and the remaining interactions are used as the test set. The items with observed interactions with the user are used as positive samples for the user, and items with unobserved interactions are randomly sampled for users as negative samples that form training pairs with each positive sample.

**Benchmarks and Evaluation Metrics.** To reveal the value of our proposed method, we compared it with several **SOTA** benchmark methods, covering **MF**-based (BPR-MF [12]), **MLP**-based (NeuMF [8]), **GCN**-based (GCMC [1], NGCF [17,22], LR-GCCF [3], LightGCN [7], IMP-GCN [13]). The all-ranking strategy used to calculate each metric and use all items that do not interact with the user as a negative sample of users. We calculate the average result of all users as the prediction result of the model and present the results of Recall@N and NDCG@N [9], where N is set to 10, 20, respectively.

**Implementation Details.** All embedding sizes of the model were fixed to 64, and the embeddings of the model were initialized using Xavier [5]. The default learning rates and mini-batches are  $1e-3$  and 1024, respectively. The temperature  $\tau$  is set to 0.1. For other parameters, we use a grid search to find the optimal values. The depth of convolutional  $L$  was within  $\{1, \dots, 6\}$ , the balance weights  $W$  were set to  $\{0.1, \dots, 0.9\}$ ,  $L_2$  normalization coefficient was searched within  $\{1e-6, \dots, 1e-2\}$ , and the node dropout rating  $P$  was also introduced within  $\{0.0, \dots, 0.9\}$  for searching within. For all benchmarks, the other parameters are set as in the original paper, and we do our best to adjust the parameters to obtain the optimal values. When the results of Recall@20 on the test set are lower than the optimal value for 50 consecutive rounds, the training is stopped.

## 5.2 Performance Comparison with the SOTA Methods (Q1 & Q2)

The detailed results of all methods are shown in Table 1. Compared with the SOTA methods, the main observations are as follows:

(1) RNT-GCN consistently exceeds all benchmarks in all datasets. Compared to the strongest benchmark IMP-GCN, RNT-GCN outperforms by an average

<sup>2</sup> <http://www2.informatik.uni-freiburg.de/~chiegler/BX/>.

**Table 1.** Overall Performance Comparison With SOTA Models.

DataSet	Kindle Store				Gowalla				ml-1m				Book-Cross			
	Recall		NDCG		Recall		NDCG		Recall		NDCG		Recall		NDCG	
Method	10	20	10	20	10	20	10	20	10	20	10	20	10	20	10	20
BPR-MF	5.50	4.16	8.62	5.31	9.72	10.62	13.73	11.71	15.09	21.03	24.17	22.87	3.73	4.53	5.72	5.03
NeuMF	3.92	2.96	6.22	3.79	7.71	7.52	11.94	8.91	14.84	21.15	23.74	22.93	2.17	2.88	3.50	3.20
GCMC	5.22	3.93	8.35	5.08	9.53	10.30	14.18	11.65	16.17	22.62	26.43	24.72	3.32	3.95	5.74	4.65
NGCF	6.29	4.68	9.88	6.00	10.68	11.27	15.55	12.71	16.64	22.18	26.54	24.47	5.46	6.15	8.47	7.02
NIA-GCN	5.39	3.95	8.60	5.13	10.47	11.09	15.39	12.54	15.56	21.52	24.81	23.46	4.01	2.86	6.55	3.89
LR-GCCF	6.34	4.75	10.00	6.11	12.12	13.26	17.22	14.66	17.08	23.23	26.73	25.19	5.62	6.41	8.70	7.26
LightGCN	6.51	4.86	10.23	6.24	12.79	14.00	18.11	15.46	17.21	23.43	26.95	25.41	5.76	6.52	9.12	7.45
IMP-GCN	<u>7.04</u>	<u>5.31</u>	<u>10.80</u>	<u>6.69</u>	<u>13.06</u>	<u>14.24</u>	<u>18.65</u>	<u>15.82</u>	<u>17.37</u>	<u>23.56</u>	<u>27.11</u>	<u>25.56</u>	<u>6.33</u>	<u>7.19</u>	<u>9.77</u>	<u>8.09</u>
RNT-sc	6.64	4.99	10.11	6.29	12.27	13.47	17.51	14.93	17.25	23.39	27.07	25.45	5.90	6.70	9.13	7.57
RNT-rsc	<b>7.96</b>	<b>6.00</b>	<b>11.94</b>	<b>7.48</b>	<b>13.19</b>	<b>14.21</b>	<b>18.78</b>	<b>15.80</b>	<b>17.66</b>	<b>24.19</b>	<b>27.48</b>	<b>26.13</b>	<b>6.79</b>	<b>7.73</b>	<b>10.15</b>	<b>8.62</b>
Imp.(%)	13.07	12.99	10.56	11.81	1.00	-	0.70	-	1.67	2.67	1.36	2.23	7.27	7.51	3.89	6.55

of 13% on the Kindle Stores dataset in Recall@10 and NDCG@10. Substantial improvements have also been made on other datasets. Experimental results confirm that our proposed method can reasonably aggregate neighbor nodes and effectively extract heterogeneous information from the user-item interaction graph, preserve the information of different types of nodes, and effectively improve the quality of user and item representations.

(2) RNT-GCN yields more improvement in smaller positions (i.e., the top 10 ranks) than in larger positions (i.e., the top 20 ranks), indicating that RNT-GCN tends to rank the relevant items higher, which means the proposed model can better adapt the real-world recommendation scenario.

(3) RNT-GCN shows advantages on sparse datasets like Kindle Store and Book-Cross. This indicates that, even in the sparse dataset, RNT-GCN can still extract more useful information than methods.

(4) RNT-rsc performs better than RNT-sc. The reason is that RNT-rsc adjusts the concatenated order of different types of embedding such that the predicted ones are matched with the same type of embedding.

### 5.3 Parameter Sensitivity Analysis (Q3)

In this section, we analyze the effects of different parameters on the RNT-GCN. Due to the trend is the same on other datasets and page limitations, only the results on the Kindle Store and the ml-1m dataset are reported.

**Balance Weights.** We adjusted the balance weights to analyze the impact of two different levels of node importance on the result. The results are reported in Fig. 4. The ARN that fuses two kinds of information can obtain better results on the Kindle Store dataset. This is because ARN can more reasonably measure the importance of different neighbor nodes and aggregate more important information. The influence of balance weights on the Kindle store is greater than that of the ml-1m dataset. We speculate that because ml-1m is a dense dataset, and the weight values assigned to different neighboring nodes are very close to each other regardless of degree normalization strategy or node-level attention with

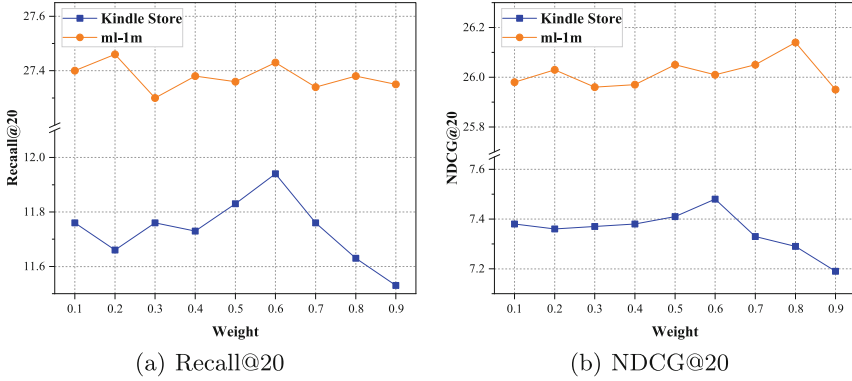


Fig. 4. Effect of Balance Weights.

normalization, the influence of fusing two different types of significance on the final results is slight. Thus, RNT-GCN results fluctuate moderately on ml-1m.

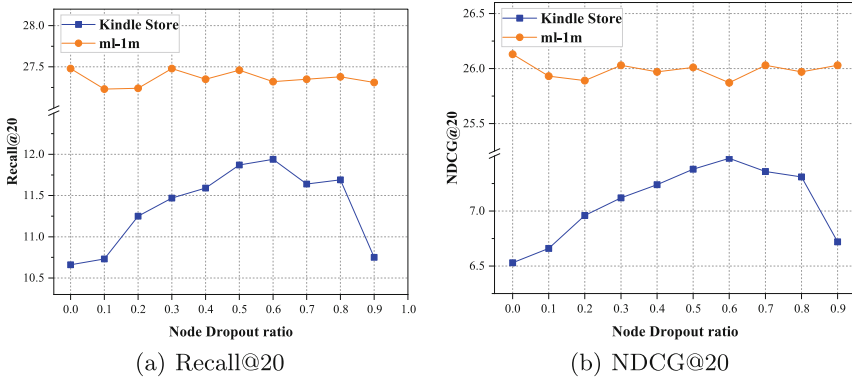


Fig. 5. Effect of Node dropout Ratio

**Node Dropout Ratio.** We adjust the node dropout ratio  $P$  to analyze the effect. The results are shown in Fig. 5. A proper node drop ratio for the Kindle Store dataset can improve performance because it effectively avoids overfitting and excessive node drop rates that lead to poor performance due to missing important information. The impact of varying node dropout ratings on the results is less significant for the ml-1m dataset, as it is denser than the Kindle store and can extract more comprehensive information from the interaction graph. The effectiveness of node dropout in improving recommended performance is varied and should be adjusted based on the dataset.

**Number of Layers.** We vary the depth of the model to investigate the effect of convolution depth on RNT-GCN. The experimental results are summarized in Table 2. When increasing the number of layers, the performance of RNT-GCN increases rapidly and comes to the peak at the 4th layer on the Kindle dataset and the 3rd layer on m1-1m dataset. With layers increasing, the performance of RNT-GCN decrease. The reason is that, more layers can obtain neighboring nodes with a larger range of perceptual fields, such that more node neighbor information can be obtained. However, when the number of convolutional layers exceeded 4 on the Kindle dataset and 3 on the m1-1m dataset, increasing the number of layers will lead to a slight decrease in performance. We hypothesize that with increasing depth, the node embeddings converge and become indistinguishable from each other.

**Table 2.** Effect of Number of Layers.

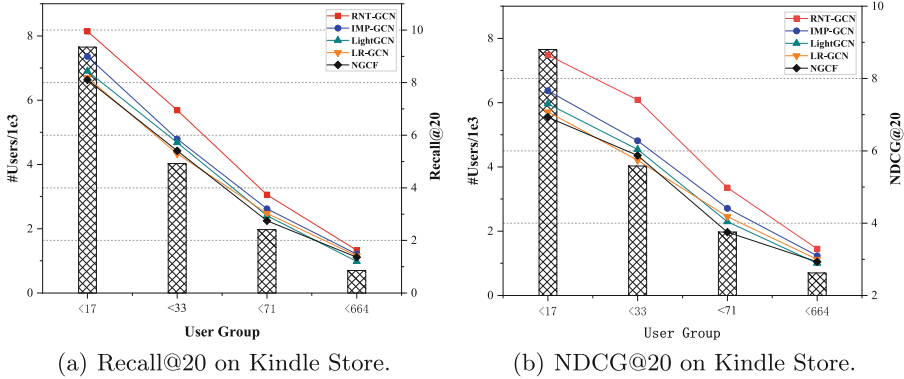
DataSet	Kindle Store				m1-1m			
Metric	Recall@10	Recall@20	NDCG@10	NDCG@20	Recall@10	Recall@20	NDCG@10	NDCG@20
1	7.30	11.01	5.47	6.84	16.57	26.03	22.76	24.73
2	7.60	11.53	5.76	7.22	17.20	26.72	23.57	25.45
3	7.61	11.45	5.77	7.34	<b>17.66</b>	<b>27.48</b>	<b>24.19</b>	<b>26.13</b>
4	<b>7.96</b>	<b>11.94</b>	<b>6.00</b>	<b>7.48</b>	17.56	27.20	24.08	25.98
5	7.74	11.61	5.88	7.31	16.87	26.49	23.39	25.30
6	7.64	11.51	5.76	7.19	16.47	25.64	23.00	24.71

#### 5.4 Data Sparsity Effect (Q4)

In this section, we investigate the efficacy of RNT-GCN in mitigating the issue of data sparsity. We divide the test set of the Kindle Store into 4 groups based on the number of interactions per user and display the Recall@20 and NDCG@20 results based on different sparsities, the detailed results are shown in Fig. 6. Due to the page limitation, we only report the results in the Kindle Store (the similar trend in other datasets). The results show that the Recall and NDCG of RNT-GCN achieve more improvement in sparse user groups than in dense user groups (i.e., the performance in the user groups with fewer than 17 and 33 times interactions gains more improvement than in the user groups with fewer than 71 and 664 times interactions). This indicates RNT-GCN is beneficial for learning the inactive users and items and can effectively alleviate data sparsity.

#### 5.5 Ablation Experiments (Q5)

To validate the effect of each part of the model, we perform ablation experiments. The results are reported in Table 3. “W/o n-i” and “W/o g-i” refer to “without graph-level importance” and “without node-level importance”, respectively. The “Cat” and “Avg” represent the pooling of embedding of each layer by concatenating and averaging, respectively. We obtained the following observations: All parts



**Fig. 6.** The comparison of user group sparsity distributions on the Kindle Store with partial baselines. The background histograms reflect the number of users for each group, while the lines depict the results.

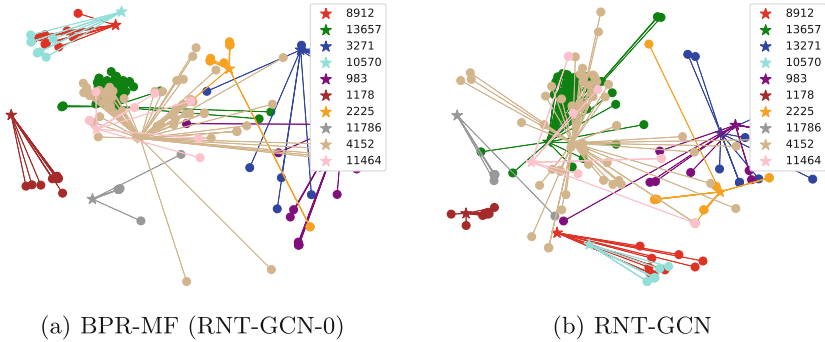
**Table 3.** Ablation experiments result on each proposed component.

DataSet	Kindle Store				ml-1m			
	Recall@10	Recall@20	NDCG@10	NDCG@20	Recall@10	Recall@20	NDCG@10	NDCG@20
W/o n-i	7.72	11.75	5.89	7.37	17.43	27.34	23.87	25.89
W/o g-i	<u>7.79</u>	<u>11.86</u>	<u>5.96</u>	<u>7.46</u>	<b>17.68</b>	<u>27.35</u>	<u>24.15</u>	<u>26.07</u>
Cat	6.47	10.05	4.84	6.16	17.13	26.82	23.23	25.17
Avg	6.93	10.64	5.21	6.57	17.50	27.08	23.86	25.69
All	<b>7.96</b>	<b>11.94</b>	<b>6.00</b>	<b>7.48</b>	<u>17.66</u>	<b>27.48</b>	<b>24.19</b>	<b>26.13</b>

of RNT-GCN prove to be workable and effective. Compared with using only the importance of the graph level or the node level, ARN that incorporates both can assign more reasonable importance to different nodes and learn higher-quality embedding. It can contribute to the RNT-GCN and improve the recommendation performance. Compared to the summation-based and concatenate-based strategies, RNT-GCN has achieved significant improvement. This shows that type-aware aggregate strategies and the information fuse layer can better capture the specific information of different nodes and effectively extract the high-order collaborative signals at different types of nodes.

## 5.6 Embedding Visualization (Q6)

To show a more intuitive understanding and comparison of RNT-GCN, we randomly select 10 users and visualize the embeddings of the user and item using t-SNE. The visualization is shown in Fig. 7. We can find that, compared with MF, items in RNT-GCN are more closely related to each other and have fewer items clustered. The results show that RNT-GCN improves the distribution of nodes in the embedding space and can obtain higher-quality embeddings.



**Fig. 7.** Visualization of the learned t-SNE transformed representations derived from MF and RNT-GCN. Each star represents a user from the Kindle Store dataset, while the points with the same color denote the relevant items.

## 6 Conclusion

In this paper, we propose a novel recommender system framework, RNT-GCN. In RNT-GCN, we first develop an attention refinement network that can combine information at both the graph-level and node-level to reasonably assign importance to nodes. Then, to preserve the specific properties of different types of nodes, we propose a type-aware aggregation strategy that aggregates embeddings of different depths by node type. Finally, to capture the relationship between user and item, we develop an information fusion layer that fuses two different types of embedding. Extensive experiments on four real datasets confirm the effectiveness of our method, and ablation experiments confirm that each part makes an important contribution.

**Acknowledgement.** This work was supported by Shanghai Science and Technology Commission (No. 22YF1401100), Fundamental Research Funds for the Central Universities (No. 22D111210, 22D111207), and National Science Fund for Young Scholars (No. 62202095).

## References

1. van den Berg, R., Kipf, T.N., Welling, M.: Graph Convolutional Matrix Completion (2018)
2. Chen, H., et al.: Graph neural transport networks with non-local attentions for recommender systems. In: WWW, pp. 1955–1964 (2022)
3. Chen, L., et al.: Revisiting graph based collaborative filtering: a linear residual graph convolutional network approach. In: AAAI, pp. 27–34 (2020)
4. Covington, P., Adams, J., Sargin, E.: Deep neural networks for YouTube recommendations. In: RecSys, pp. 191–198 (2016)
5. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS, pp. 249–256 (2010)

6. Gori, M., Pucci, A.: ItemRank: a random-walk based scoring algorithm for recommender engines". In: IJCAI, pp. 2766–2771 (2007)
7. He, X., et al.: LightGCN: simplifying and powering graph convolution network for recommendation. In: SIGIR, pp. 639–648 (2020)
8. He, X., et al.: Neural Collaborative Filtering. In: WWW, pp. 173–182 (2017)
9. He, X., et al.: TriRank: review-aware explainable recommendation by modeling aspects. In: CIKM, pp. 1661–1670 (2015)
10. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: ICLR (2015)
11. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
12. Koren, Y., Bell, R.M., Volinsky, C.: Matrix factorization techniques for recommender systems. *Computer* **8**, 30–37 (2009)
13. Liu, F., et al.: Interest-aware message-passing GCN for recommendation. In: WWW, pp. 1296–1305 (2021)
14. Mao, K., et al.: UltraGCN: ultra simplification of graph convolutional networks for recommendation. In: CIKM, pp. 1253–1262 (2021)
15. Mehta, N., Pacheco, M.L., Goldwasser, D.: Tackling fake news detection by continually improving social context representations using graph neural networks. In: ACL, pp. 1363–1380 (2022)
16. Rendle, S., et al.: BPR: bayesian personalized ranking from implicit feedback. In: UAI, pp. 452–461 (2009)
17. Sun, J., et al.: Neighbor interaction aware graph convolution networks for recommendation. In: SIGIR, pp. 1289–1298 (2020)
18. Sun, J., et al.: Separated graph neural networks for recommendation systems. In: IEEE TII, pp. 382–393 (2023)
19. Velickovic, P., et al.: Graph attention networks. In: ICLR (2018)
20. Wang, X., et al.: Disentangled graph collaborative filtering. In: SIGIR, pp. 1001–1010 (2020)
21. Wang, X., et al.: KGAT: knowledge graph attention network for recommendation. In: KDD, pp. 950–958 (2019)
22. Wang, X., et al.: Neural graph collaborative filtering. In: SIGIR, pp. 165–174 (2019)
23. Xu, K., et al.: Representation learning on graphs with jumping knowledge networks. In: ICML, pp. 5449–5458 (2018)
24. Zhang, Y., et al.: Geometric disentangled collaborative filtering. In: SIGIR 2022, Madrid, Spain, pp. 80–90 (2022)