Pre-training Recommender Systems via Reinforced Attentive Multi-relational Graph Neural Network

Xiaohan Li†, Zhiwei Liu†, Stephen Guo‡, Zheng Liu†, Hao Peng§, Philip S. Yu†, Kannan Achan‡
†University of Illinois at Chicago, Chicago, IL, USA
{xli241, zliu213, zliu212, psyu}@uic.edu
‡Walmart Global Tech, San Francisco Bay Area, CA, USA
{SGuo, KAchan}@walmartlabs.com
§Beihang University, Beijing, China
penghao@act.buaa.edu.cn

2021 IEEE International Conference on Big Data (Big Data) | 978-1-6654-3902-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/BigData52589.2021.9671830

***Abstract*—Recently, Graph Neural Networks (GNNs) have proven their effectiveness for recommender systems. Existing studies have applied GNNs to capture collaborative relations in the data. However, in real-world scenarios, the relations in a recommendation graph can be of various kinds. For example, two movies may be associated either by the same genre or by the same director/actor. If we use a single graph to elaborate all these relations, the graph can be too complex to process. To address this issue, we bring the idea of pre-training to process the complex graph step by step. Based on the idea of divide-and-conquer, we separate the large graph into three sub-graphs: user graph, item graph, and user-item interaction graph. Then the user and item embeddings are pre-trained from user and item graphs, respectively. To conduct pre-training, we construct the multi-relational user graph and item graph, respectively, based on their attributes.**

In this paper, we propose a novel Reinforced Attentive Multi-relational Graph Neural Network (RAM-GNN) to pre-train user and item embeddings on the user and item graph prior to the recommendation step. Specifically, we design a relation-level attention layer to learn the importance of different relations. Next, a Reinforced Neighbor Sampler (RNS) is applied to search the optimal filtering threshold for sampling top-k similar neighbors in the graph, which avoids the over-smoothing issue. We initialize the recommendation model with the pre-trained user/item embeddings. Finally, an aggregation-based GNN model is utilized to learn from the collaborative relations in the user-item interaction graph and provide recommendations. Our experiments demonstrate that RAM-GNN outperforms other state-of-the-art graph-based recommendation models and multi-relational graph neural networks.

***Index Terms*—recommender system, graph neural network, reinforcement learning**

I. Introduction

With the development of the Internet in recent years, the information explosion problem has become an inevitable issue to consider when designing a large-scale online platform. The overload of information hinders users' ability to find what they really need among a large amount of items. To enhance users' experience, recommender systems have been applied in many web applications including e-commerce [1], [2], social recommendations [3], [4], and movie recommendations [5].

In order to improve performance, Knowledge Graph (KG)-based recommender systems have achieved more consideration recently due to KG's capacity of unifying user-item interactions and their side information in a graph [6]–[11]. However, since the KGs used for recommender systems contain multiple kinds of relations and are generally dense, directly conducting aggregation on KGs would suffer from the over-smoothing issue [12] when learning node embeddings. To be more specific, a GNN model aggregates the neighbor information into the central node. However, too complex neighbor information may contain more noises, which thus impedes the aggregation of a GNN model to retrieve relevant information. Also, when we directly apply GNNs on KGs, GNNs are not able to explore high-order connectivity because the number of neighbors grows exponentially w.r.t. the number of layers. Take the movie dataset as an example, when two users share similar ages and two movies have the same director, the connection between user A and movie Y is:

- user A → age → user B → movie X → director → movie Y

which is a long path with multi-hop connection that GNNs can hardly explore.

The spirit of *divide-and-conquer* motivates us to leverage the pre-training and fine-tuning paradigm to learn user/item embeddings. The key idea of pre-training is to learn from other types of data, e.g., data from other sources or data for other tasks, and generate the embeddings [13]–[15]. In order to preserve the high-order information while avoiding the over-smoothing issue, we first pre-train the model using various user-user and item-item relations in the KG, and then fine-tune it only using the collaborative relation. In the pre-training step, the model can learn from the content of users and items and generate embeddings containing the information of various relations. Moreover, in the fine-tuning step, another model emphasizes on the recommendation task, which only incorporate the collaborative information to achieve the best performance.

Based on the idea above, we split the long path into several shorter paths which fit the characteristics of GNNs. We divide the complex knowledge graph into three sub-graphs: user-user graph, item-item graph, and user-item interaction graph. In the

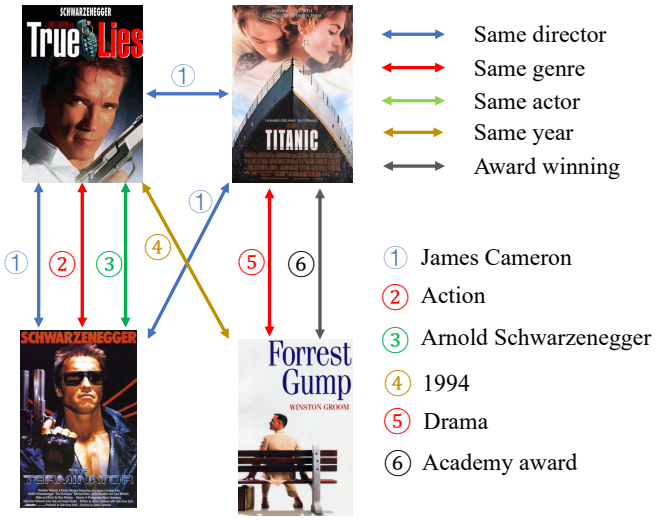


Fig. 1. Multi-relational item graph used in the pre-training step. Items sharing the same attributes are connected with an edge. In the graph, item attributes include two parts: relation types (edges) and relation values (numbers). There can be more than one edge between a pair of items.

pre-training step, we construct the multi-relational user graph and item graph, where Figure 1 shows an example of item graph. In these graphs, nodes are connected when they share the same features, so there can be multiple relations between a pair of nodes. In the user-item interaction graph, we utilize the user and item embeddings learned in the pre-training step as the initial embeddings and fine-tune them via a GNN.

However, it is challenging to adopt the aforementioned paradigm on the complex recommendation data. One challenge is **uneven relation importance**, which means different types of relations in the graph matter unevenly when learning the node embeddings. Figure 1 shows an example of this unevenness. In Figure 1, there are five types of relations shown in the graph, and some of them are more important when measuring the similarity between movies. For example, the similarity between *True Lies* and *Titanic* is higher than that between *True Lies* and *Forrest Gump*, since a common director is a stronger association than the same year of movies. Therefore, it is required to characterize the importance of those relations. The pre-training model needs to learn the importance score of different relations.

The other challenges is **uneven distributions**, which indicates the distributions and statistics vary w.r.t. relations. For a particular node, some relations in Figure 1 connect few neighbors for this node, but for some other relations it may have thousands of neighbors. For example, one movie can only have several neighboring movies sharing the same director, but may have millions of movies sharing the same genre. Therefore, the latter relation extremely undermine the uniqueness of the movie when conducting aggregation in the GNN because of the over-smoothing problem. Thus our model should learn from these neighbors appropriately and select the most useful ones to train the model.

To solve the challenges above, we propose a novel model named **R**einforced **A**ttentive **M**ulti-relational Graph Neural Network (RAM-GNN). First, we construct a user graph and an item graph with multiple types of relations. Unlike most existing multi-relational graphs [16]–[18], our user and item graphs have multiple relation between a pair of nodes. Additionally, the relations have values, such as the director of the movie in the "share directors" relation. In this case, we model the meta structure of the multi-relational graph as a quadruplet, which is "item - (relation type) - (relation attribute) - item", to learn the item embeddings as well as the relation embeddings. In RAM-GNN, we apply two modules to solve the two challenges above and learn the node and relation embeddings. One is relation-level attention, which learns the correlations between relations and items. The other is Reinforced Neighbor Sampler (RNS), which samples the top-k similar nodes to filter less relevant neighbors and enhance the quality of learned embeddings via an adaptive Reinforcement Learning (RL) process. To optimize the filtering threshold for each relation, RNS tries to find a trade-off between the average neighbor distances and the total number of the sampled nodes. The model is trained by a GNN loss and a similarity loss jointly with unsupervised learning. We pre-train the RAM-GNN model on the user graph and item graph respectively to learn user and item embeddings. Then another GNN initialized by the pre-trained embeddings is applied to fine-tune the embeddings and provide recommendation results. In experiments, we demonstrate that our model outperforms other state-of-the-art methods.

In this paper, we summarize our contributions as follows:

- We propose the RAM-GNN model to learn the embeddings of users and items for a recommender system. The two main components: relation-level attention and reinforced neighbor sampler provide robust and efficient embedding learning.
- We design another GNN to further learn the user and item embeddings based on pre-trained item/user and relation embeddings. This GNN provides a list of items for the user as recommendation results.
- We conduct experiments on two real-world datasets for the recommendation task. The experiments demonstrate the effectiveness of the RAM-GNN compared to eight state-of-the-art baselines.

II. Preliminaries

Graph Neural Networks (GNNs) are neural networks that can process graphs directly. By aggregating neighboring information, GNNs can extract structural knowledge from a graph to learn node embeddings. Due to its capacity to process unstructured data, GNNs are widely used on many tasks such as node classification, graph classification, and link prediction.

The forward propagation procedure of a GNN on graph $G(\mathcal{V}, \mathcal{E})$ is to update the representation of each node $v_i \in \mathcal{V}$ through neighboring nodes. Suppose for each node i there is an initial node representation $\boldsymbol{h}_i^{(0)}$ as the input of the model. Then, each hidden layer of GNN learns the central node

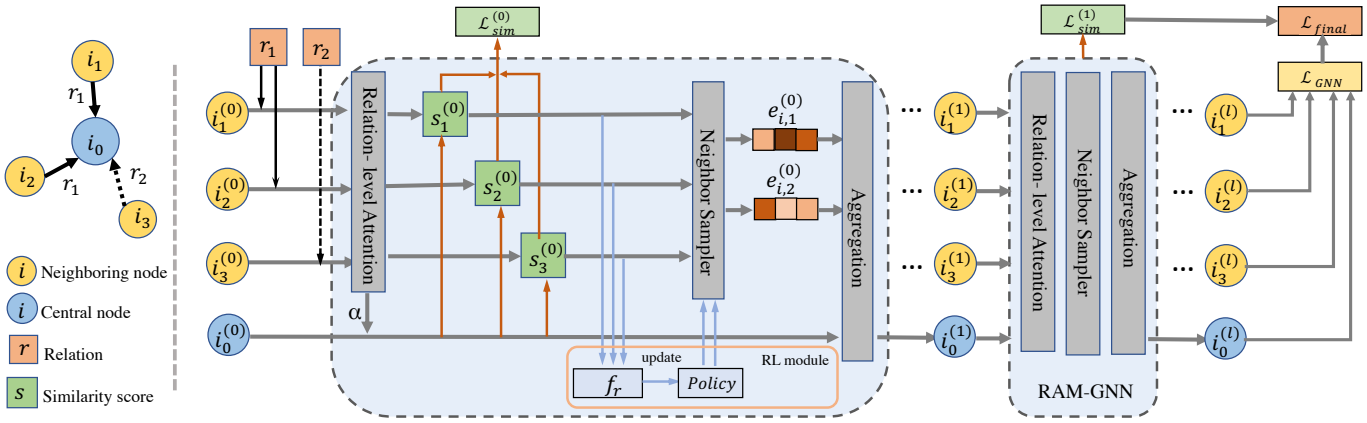


Fig. 2. The model structure of RAM-GNN. The central node i_0 is connected by neighboring nodes i_1, i_2 and i_3 with relations r_1 and r_2. Relation-level attention learns the importance of different types of relations. Reinforced Neighbor Sampler finds the most similar neighbors to reduce redundant information and improve efficiency.

TABLE I
NOTATIONS

Notation	Explanation
$\mathbf{e}_i, \mathbf{e}_t, \mathbf{e}_v, \mathbf{e}_j$	The embeddings of head item, relation type relation value, and tail item
$\mathbf{e}_n$	The combination embedding of $\mathbf{e}_t$ and $\mathbf{e}_v$
ϕ	Composition operation
σ	Activation function
$\mathcal{N}(j)$	All neighbors of item j
$\boldsymbol{Aggr}(\cdot)$	Aggregation function
$\mathbf{W}_{\text{key}}, \mathbf{W}_{\text{qry}}, \mathbf{W}_{\text{val}}$	The weight matrices of key,query, and value
α_{in}	The relation-level attention value
$\|$	Concatenation operation
$d(\mathbf{e}_i, \mathbf{e}_j), s(\mathbf{e}_i, \mathbf{e}_j)$	distance and similarity between node i and j
ϵ	the change of threshold in each iteration
AND	Average neighbor distance
γ, Γ	iteration, maximum iteration
$f_r(\cdot)$	reward function
$\mathcal{L}_{GNN}, \mathcal{L}_{sim}, \mathcal{L}_{final}$	GNN, similarity and final loss of RAM-GNN
$\mathbf{y}_j, \hat{\mathbf{y}}_j$	Real value and prediction value
$y_{i,j}$	The ground truth label of node i and j
$\mathbf{x}_u, \mathbf{x}_i$	The user and item embedding in recommendation

representation $\boldsymbol{h}_i^{(l)}$ from the previous hidden layer $\boldsymbol{h}_i^{(l-1)}$ by aggregating the neighboring nodes as follows:

$$\boldsymbol{h}_i^{(l)} = \sigma\Big(\boldsymbol{W}^{(l)}(\underset{j\in\mathcal{N}(i)}{\boldsymbol{Aggr}}(\boldsymbol{h}_j^{(l-1)}) \oplus \boldsymbol{h}_i^{(l-1)})\Big), \tag{1}$$

where σ is the non-linear activation function such as LeakyReLU and $\mathcal{N}(i)$ represents the set of neighbors of node i in the graph. The neighborhood aggregation function $\boldsymbol{Aggr}(\cdot)$ sums neighboring information up and applies an activation function (*e.g.*, sigmoid or LeakyReLU). $\oplus$ represents the combination operation of aggregated neighbor embedding and the central node embedding from the previous layer, e.g., concatenation operation.

When the number of edge types is more than one, we call this kind of graph as multi-relational graph. A classic method [17] to apply GNN to multi-relational graph is Relational GCN (R-GCN). It re-writes the GNN formula as:

$$\boldsymbol{h}_i^{(l)} = \sigma\Big(\boldsymbol{W}_r^{(l)}(\sum_{r\in\mathcal{R}}\underset{j\in\mathcal{N}_r(i)}{\boldsymbol{Aggr}_{(r)}}(\boldsymbol{h}_j^{(l-1)}) \oplus \boldsymbol{h}_i^{(l-1)})\Big), \tag{2}$$

where $\boldsymbol{Aggr}_{(r)}, \mathbf{W}_r$ and $\mathcal{N}_r(i)$ denote the aggregation function, weight matrix and the neighbors of node i corresponding to the relation r. $\mathcal{R}$ is the set of all relations. However, this method suffers from over-parameterization because each relation is associated with a weight matrix. In this paper, our model tries to solve this problem with entity-relation composition operations to reduce the number of weight matrices in R-GCN to 1.

In the following sections, we first present how to pre-train the item graph with our proposed RAM-GNN. Then, we illustrate how to apply the knowledge learned in the pre-training process to help provide recommendations.

III. Model

To overcome the over-smoothing problem and find the most relevant information in the data, we propose the RAM-GNN model to pre-train the user and item embeddings on the user and item graphs, respectively. We use the item graph in Figure 1 as an example to help understand the pre-training paradigm. The user embeddings are pre-trained in an analogous way upon the user graph.

A. Multi-relational Graph

In real-world applications, there exist multiple relations between a pair of items that have concrete semantics. As shown in Figure 1, the relations between items form a multi-relational graph. Given an item pair (i, j), the relations between them are defined as a set of $r =< t, v >$, where t denotes the relation type and v is the relation value. Our goal is to integrate all these relations to learn the item embeddings. Different from the triplets in traditional KGs, the meta structure of the item graph is a quadruplet (i, t, v, j), where i and j are the head and tail item entities, t is the relation type, and v is the relation value. Motivated from methods used in KG embeddings [16], we propagate the message from tail item j to head item i through relation r, and the process is formulated as:

$$\mathbf{e}_i = \phi(\mathbf{e}_j, \mathbf{e}_t, \mathbf{e}_v), \tag{3}$$

where ϕ is a composition operator, $\mathbf{e}_i, \mathbf{e}_t, \mathbf{e}_v$ and $\mathbf{e}_j \in \mathbb{R}^d$ denote the embeddings of head item, relation type, relation value, and tail item respectively in the KG. Different from traditional KG algorithms, which are designed for triplets, our method learns from the quadruplets and composition operations to generate embeddings. For relation types and relations values, we combine them together into integrated embeddings. Here, we concatenate the embeddings of relation types and relations values as:

$$\mathbf{e}_n = \mathbf{e}_t \parallel \mathbf{e}_v, \tag{4}$$

where $\parallel$ is the concatenation operation. In Section 2.3, we will use $\mathbf{e}_v$ in the recommendation step. Then, we can re-write the Eq. 3 as:

$$\mathbf{e}_i = \phi(\mathbf{e}_j, \mathbf{e}_n), \tag{5}$$

where $\mathbf{e}_n$ is the new embedding that is composed of the relation type $\mathbf{e}_t$ and the relation value $\mathbf{e}_v$. Three kinds of composition operations ϕ are applied in our model:

- Addition: $\mathbf{e}_i = \mathbf{e}_j + \mathbf{e}_n$,
- Multiplication: $\mathbf{e}_i = \mathbf{e}_j * \mathbf{e}_n$,
- Circular-correlation: $\mathbf{e}_i = \mathbf{e}_j \star \mathbf{e}_n$.

Here, the three operations are motivated from TransE [16], DistMult [19], and HolE [20]. The performance of these operations are discussed in Section 3.5.1.

To learn item embeddings based on the quadruplets, we design our model **R**einforced **A**ttentive **M**ulti-relational Graph Neural Network (RAM-GNN), which are displayed in Figure 2. In this model, relation-level and node-level attention layers are proposed to handle the multi-relational graph. Details are introduced in the following sections.

B. Relation-level Attention

For all relations associated with the item i, they may contribute unequally in learning the item embedding. For example, the genres and actors of a movie have different importance. Additionally, thousands of movies may share the same genre, while only a few movies may have common actors. To solve this **uneven relation importance** challenge, we propose the relation-level attention. The attention mechanism [21] has been widely adopted in existing deep learning models to infer the importance of inputs. In our problem, we adopt the attention mechanism into the multi-relational graph to calculate the attention weights for each relation. We re-write the entity-relation composition operation based on self-attention:

$$\mathbf{a}_{jn} = \mathbf{p}^T \sigma(\mathbf{W}_{\text{key}}\mathbf{e}_j + \mathbf{W}_{\text{qry}}\mathbf{e}_n + \mathbf{b}), \tag{6}$$

$$\alpha_{jn} = \text{Softmax}(\mathbf{a}_{jn}), \tag{7}$$

$$\mathbf{e}_i = \alpha_{jn}\mathbf{W}_{\text{val}}\phi(\mathbf{e}_j, \mathbf{e}_n), \tag{8}$$

where $\mathbf{p}, \mathbf{b} \in \mathbb{R}^d$, and α_{jn} is the relation-specific attention weight for item pair (e_i, e_j) and relation embedding e_n. $\mathbf{W}_{\text{key}}, \mathbf{W}_{\text{qry}}, \mathbf{W}_{\text{val}}$ are key, query, and value weight matrices in the self-attention [21], respectively. With relation-specific attention, we can learn how the relation influences the head item and generate the embedding of tail item j considering their correlation.

C. Reinforced Neighbor Sampler

After the relation-level attention module, we propose the Reinforced Neighbor Sampler (RNS) based on Reinforcement Learning (RL). RNS searches the top-k similar items to reduce redundant information and improve efficiency. In the item or user graph, there can be massive numbers of neighbors for a node. For example, thousands of items can share the same category, and near half of the users are of the same gender. As a result, if we aggregate all neighbors together to learn the embedding of the target node, the unique features of the target node will be overwhelmed in the massive neighbors, which is the over-smoothing issue [12]. To deal with **uneven distributions** challenge, we design an adaptive neighbor sampler to prune the irrelevant neighbors and only select the top-k similar neighboring items to perform aggregation. Given the difference of relations in the graph, the filtering thresholds of neighbor selections are difficult to be pre-defined as hyper-parameters. Motivated by [11], [22]–[25], our proposed RNS searches the optimal threshold for each relation via an RL process.

We measure the item similarity based on negative sampling [11], [26]. If two items are connected by an edge in the item graph, we take this pair of items as a positive instance. For each item, we also randomly select several irrelevant items as negative samples. We measure the item similarity by minimizing the distance of the positive item pairs and maximizing the distance of negative pairs. Inspired by [27], Multi-Layer Perceptron (MLP) is applied to measure the distance of items in RNS. Formally, the distance between two items is as follows:

$$d(\mathbf{e}_i, \mathbf{e}_j) = \parallel \sigma(MLP(\mathbf{e}_i)) - \sigma(MLP(\mathbf{e}_j)) \parallel_1 \tag{9}$$

where σ is the activation function, and we choose sigmoid in the above equation. i is the target item, and j is the positive or negative samples w.r.t. i. The output of the MLP layer is a scalar. With the sigmoid activation, the distance is projected to $(0, 1)$, where a closer distance represents two items are stronger similar. To measure the similarity of a pair of items and simplify calculation, we convert distances to similarity scores as:

$$s(\mathbf{e}_i, \mathbf{e}_j) = 1 - d(\mathbf{e}_i, \mathbf{e}_j), \tag{10}$$

where $s(\mathbf{e}_i, \mathbf{e}_j)$ is the similarity score of item i and j, and a higher similarity score means i and j are more similar. We use the similarity score to discriminate whether the two items are similar or not.

Based on the item similarity, we define a cross-entropy based similarity loss as the loss function of MLP layers. The similarity loss is a supplement of the GNN loss, which is formulated as:

$$\mathcal{L}_{sim} = \sum_{j \in \mathcal{I}} -\log y_{i,j} \cdot s(\mathbf{e}_i, \mathbf{e}_j), \tag{11}$$

where $\mathcal{I}$ is an item set containing all positive and negative samples w.r.t. item i, and $y_{i,j}$ is the ground truth to determine whether j is a positive or negative sample. With the similarity

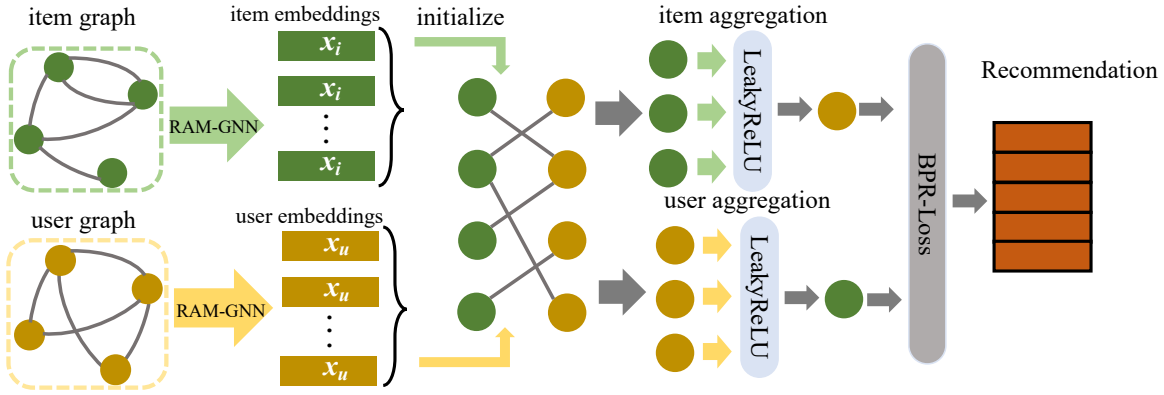


Fig. 3. The complete pre-training framework. RAM-GNN pre-trains user and item embeddings on the user and item graph. Then, the learned embeddings are used to initialize the node embeddings in the interaction graph. With another aggregation-based GNN, we obtain the recommendation results through BPR loss.

loss, we can learn the similarity score for each item pair and rank them decreasingly to find the most similar ones.

In the item graph, there are multiple types of relations. The number of items we select to conduct aggregation should vary w.r.t. different relations. Therefore, we adopt filtering thresholds for all relations. However, incorporating those thresholds as hyper-parameters is time-consuming because the number of relations can be extremely large.

To improve efficiency in the training process, we propose Reinforced Neighbor Sampler (RNS) to automatically search for the optimal filtering threshold for each type of relation during the training process. We express the RL process of finding filtering thresholds as Bernoulli Multi-armed Bandit (BMAB) $\mathcal{B}(\mathcal{A}, \mathcal{R}, \mathcal{T})$ [28], which is a simplified version of Markov decision process as there is no state in it. In the BMAB, $\mathcal{A}, \mathcal{R}$ and $\mathcal{T}$ represent the action space, the reward function, and the terminal condition, respectively. Initially, we set the k_t as the filtering thresholds of the neighbor sampler corresponding to a relation type t. Then we utilize the reward function f_r to determine the action of increasing or decreasing the k_t. Specifically, the BMAB process are defined as follows:

Action space. The action represents how we adjust the k_t based on the reward function. We define a fixed small value ϵ as the action. k_t is increased or decreased by ϵ to find the optimal filtering thresholds.

Reward function. We design the reward function to determine increasing or decreasing the k_t based on the distance of items we define in Eq. 9. The reward function discovers the most similar items iteratively to achieve the minimum Average Neighbor Distance (AND) with as large a filtering threshold as possible. The AND in iteration γ is calculated as follows:

$$AND^{\gamma} = \frac{\sum_{j \in \mathcal{N}_{k_t}(i)} d(\mathbf{e}_i, \mathbf{e}_j)^{\gamma}}{|\mathcal{N}_{k_t}(i)|}, \tag{12}$$

where $\mathcal{N}_{k_t}(i)$ is the top k_t neighbor items of item i. It is a trade-off between the AND and filtering threshold k_t, since we intend to have more neighbors into aggregation and less AND. Based on this idea, we can adjust the AND in the reward function as:

$$f_r(AND^{\gamma}) = \begin{cases} +1, & AND^{\gamma-1} \geq AND^{\gamma}, \\ -1, & AND^{\gamma-1} < AND^{\gamma}, \end{cases} \tag{13}$$

where f_r is the reward function. If the output of f_r is positive, k_t is increased by ϵ and vice versa. In this way, a positive output of f_r leads to a smaller AND compared to last iteration so that we can increase k_r to have more neighbors involved. In opposite, if the f_r is negative, we need to decrease k_r to limit the AND. These two actions take place back and forth until reaching a convergence.

Termination condition. The RL process converges when the AND does not vibrate explicitly during the training iteration, which leads to the termination condition as:

$$\Big| \sum_{\gamma-10}^{\gamma} f_r(AND^{\gamma}) \Big| \leq \epsilon, \text{ where } \gamma > 10. \tag{14}$$

As this inequality represents the cumulative reward of the recent ten training iterations is less than ϵ, we claim the RL converges under this termination condition. When the termination condition has reached, we fix the filtering threshold in the following GNN training iterations.

D. RAM-GNN

To combine the relation-level attention and reinforced neighbor sampler into a unified framework, we propose our Reinforced Attentive Multi-relational Graph Neural Network (RAM-GNN) to generate node embeddings. After the RNS module, we select the top-k similar neighbors for each item, so the next step is to aggregate all the filtered neighbors to learn the item embeddings. Since we have taken the relation type into account in the relation-level attention module, the aggregation process is defined as:

$$\mathbf{e}_i^{(l)} = \sigma\Big(\alpha_{jn} \mathbf{W}_{\text{val}}^{(l)} \big(\underset{j \in \mathcal{N}_{k_t}(i)}{\boldsymbol{Aggr}} \left(\phi(\mathbf{e}_j^{(l-1)}, \mathbf{e}_n^{(l-1)})\right) \oplus \mathbf{e}_i^{(l-1)} \big)\Big), \tag{15}$$

where l is the number of layer in GNN and j is the selected neighbors from RNS. The aggregation layer can be stacked layer-to-layer and construct a deep neural network.

The loss function of the RAM-GNN is composed of two parts: GNN loss and similarity loss. The GNN loss serves for training all parameters in the model, while the similarity loss is mainly designed for helping RNS. The GNN is also trained via negative sampling, and the loss function is:

$$\mathcal{L}_{GNN} = \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{I}} - \log y_{i,j} \cdot s(\mathbf{e}_i, \mathbf{e}_j), \tag{16}$$

where $\mathcal{V}$ is the set of all nodes in the graph as we need to go through the whole graph to generate node embeddings. The final loss function of RAM-GNN is the combination of GNN loss, similarity loss, and regularization terms, whose formula is:

$$\mathcal{L}_{final} = \mathcal{L}_{GNN} + \lambda_\gamma \sum_{\gamma=1}^{\Gamma} \mathcal{L}_{sim}^{\gamma} + \lambda_1 \|\Theta_1\|, \tag{17}$$

where $\lambda_\gamma, \lambda_1$ are weight hyperparameters and Θ is all parameters in the RAM-GNN. Γ is the iteration number when the RL process converges. All weight matrices in the the pre-training model are randomly initialized and learned in an end-to-end back-propagation training paradigm.

E. Recommendation

After learning the pre-trained item embeddings, we further leverage the user-item interaction graph to fine-tune the user/item embeddings via a GNN. Motivated by [29], we learn the user/item embeddings through layer-to-layer aggregation to incorporate the collaborative relation. Different from previous models, we utilize the learned relation embeddings of the items from the pre-training process. In the aggregation process, we concatenate the relation value embeddings together with the item embeddings to learn a better representation of items.

The structure of the recommendation framework is shown in Figure 3. The first-order propagation aggregates the item embeddings into user embeddings:

$$\mathbf{x}_i^{(0)} = \mathbf{W}_3(\mathbf{e}_i \parallel \mathbf{e}_v), \tag{18}$$

$$\mathbf{x}_u^{(l)} = \sigma\Big(\mathbf{W}_4^{(l)}(\underset{i \in \mathcal{N}(u)}{\boldsymbol{Aggr}}(\mathbf{x}_i^{(l-1)}) \oplus \mathbf{x}_u^{(l-1)})\Big), \tag{19}$$

where $\mathbf{x}_i, \mathbf{x}_u \in \mathbb{R}^d$ represent item and user embeddings, respectively. $\mathbf{W_3} \in \mathbb{R}^{d \times d'}, \mathbf{W_4} \in \mathbb{R}^{d \times d}$ are both trainable weight matrices and d' is the length of concatenated embeddings. $\parallel$ is the concatenation operation to combine the relation value embeddings with the pre-trained item embeddings. Note that the number of $\mathbf{e}_v$ may be more than one. As such, we concatenate the embeddings of all relation values. With first-order propagation, the user embeddings can incorporate the embeddings of interacted items and their attributes.

TABLE II
THE NUMBER OF USERS, ITEMS, INTERACTIONS, AND RELATION TYPES IN THE DATASETS. 'USER REL.' AND 'ITEM REL.' REPRESENT THE NUMBER OF RELATION TYPES OF USERS AND ITEMS, RESPECTIVELY.

Datasets	Items	Users	Interactions	User Rel.	Item Rel.
MovieLens	1,682	943	100,000	3	4
KKBox	24,613	61,877	2,170,690	2	4

The second-order propagation aggregates the user embeddings into item embeddings. The process is similar to the item side aggregation, which is:

$$\mathbf{x}_u^{(0)} = \mathbf{W}_3(\mathbf{e}_u \parallel \mathbf{e}_v), \tag{20}$$

$$\mathbf{x}_i^{(l)} = \sigma\Big(\mathbf{W}_4^{(l)}(\underset{i \in \mathcal{N}(u)}{\boldsymbol{Aggr}}(\mathbf{x}_u^{(l-1)}) \oplus \mathbf{x}_i^{(l-1)})\Big), \tag{21}$$

In this way, we can further fine-tune the pre-trained embeddings with the collaborative relation to make the embeddings more comprehensive. After the second-order propagation, Eq. 19 and Eq. 21 can be stacked interchangeably to build a deeper neural network and explore higher-order proximity.

Based on user and item embeddings, we utilize a widely used BPR loss [30] to train the model. The objective function is as follows:

$$\hat{y}_{u,i} = \mathbf{x}_i^T \mathbf{x}_u \tag{22}$$

$$\mathbb{L}(y_{u,i}, \hat{y}_{u,i}) = -\ln \sigma(y_{u,i} - \hat{y}_{u,i}) + \lambda_2 \|\Theta_2\|, \tag{23}$$

where Θ_2 is all trainable parameters in the GNN model, $\|\Theta_2\|$ is the L_2 normalization of all trainable parameters, and λ is the regularization coefficient. The recommendation process is also trained by the back-propagation algorithm.

IV. EXPERIMENTS

In this section, we conduct experiments on two real-world datasets to evaluate the performance of our proposed RAM-GNN model and the whole pre-training framework. We aim to answer the following research questions:

- **RQ1**: How does the whole pre-training framework perform compared with other recommender system algorithms?
- **RQ2**: How does RAM-GNN perform compared with state-of-the-art multi-relational GNN models?
- **RQ3**: What is the influence of the relation-level attention and reinforced neighbor sampler?
- **RQ4**: How does RNS perform in finding the optimal filtering threshold?
- **RQ5**: How do we select suitable hyperparameters (e.g., choices of composition operations, embedding dimensions) when training the model?

A. Experimental Settings

1) Datasets: We conduct our experiments on two public datasets: MovieLens[1] and KKBox[2].

[1]https://grouplens.org/datasets/movielens/

[2]https://www.kaggle.com/c/kkbox-music-recommendation-challenge/data

- **MovieLens.** It is a widely used benchmark dataset published by GroupLens [31]. The user relation types we used in this dataset are ages, genders and occupations. For item relation types, we use genres, directors, movies, and released years. The graph we build to pre-train the item graph is similar to Figure 1. The user ratings are transformed into binary numbers to indicate implicit feedback. We utilize users' ages and occupations to construct the user graph.
- **KKBox.** This dataset was first introduced in the WSDM Cup 2018 Challenge[3]. It contains four types of relations, including genres, artists, composers and lyricists. We also use ages and living cities as user features.

The statistics of these two datasets are listed in Table II.

2) Evaluation Metrics: Two evaluation metrics are used to measure the performance of our RAM-GNN framework:

HR. This is the Hit Ratio (HR) of target items that are in the recommendation lists.

MRR. We can measure the performance of the model w.r.t the ranking list of items. Suppose that the model produces a list of items to the user, and the list is ordered by the confidence of the prediction. In this case, a higher MRR score means target items tend to have higher rank positions in the predicted item lists.

NDCG. This measures the ranking quality. It normalizes the Discounted Cumulative Gain (DCG) to be between 0 and 1 by Ideal Discounted Cumulative Gain (IDCG).

3) Implement Settings: We implement our model in PyTorch [4]. The number of layers in RAM-GNN is set to 2, and in the recommendation GNN, it is set to 4, according to [29]. As will see later, with the help of pre-training, RAM-GNN with a fewer number of layers on a smaller user-item interaction graph outperforms state of the art GNN based recommendation methods.

B. Recommendation Performance (RQ1)

To prove the superiority of our proposed RAM-GNN framework, we conduct experiments on two datasets and compare our model with eight baseline methods. In this section, we denote the pre-training and recommendation steps together as the RAM-GNN framework.

1) Compared Methods: We compare the performance of our model with the following eight baselines:

- **MF-BPR** [30]: This is matrix factorization with Bayesian Personal Ranking (BPR) as the loss function.
- **FM** [32]: Factorization Machine (FM) is a content-based model which uses feature interactions to model user preferences. We take the side information as additional input features in both datasets.
- **NFM** [27]: Neural Factorization Machine applies an MLP to model the high-order feature interactions.
- **FISM** [33]: This is an item collaborative filtering (ICF) model which learns the user embedding by aggregating the item embeddings that she has interacted with.
- **CKE** [5]: It models the items with a knowledge graph that connects items with their features and then learns the embeddings of items that can be used in CF models.
- **LightGCN** [34]: It simplifies GNN based recommender systems like NGCF [27] to make them linear models and achieve better performance.
- **KGPolicy** [11]: Knowledge Graph Policy Network is to explore high-quality negatives via reinforcement learning. We take features of users and items as nodes in the knowledge graph.
- **RCF** [35]: Relational Collaborative Filtering (RCF) considers the multi-relational pattern in the items. It utilizes the attention mechanism to combine the multi-relational pattern with the collaborative relation.

2) Results: The experimental results are presented in Table III. From these results, RAM-GNN clearly outperforms all baselines. We summarize the following observations.

- Compared with RCF, our model achieves better performance on both datasets. RCF is the state-of-the-art recommendation model among all baselines, primarily because it considers the multi-relational pattern in the data. Our model can not only take multi-relational patterns into consideration but also incorporate the graph structure to build the connections at the user-level and item-level.
- KGPolicy achieves the best performance compared to all baselines except RCF and RAM-GNN. This proves that the KG can improve the results. However, considering multi-relational patterns is more important than only modeling on the KG because the performance of RCF is better than KGPolicy. Also, KGPolicy may suffer from the over-smoothing problem, which can hinder its performance.
- Our model has approximately 10% improvement over LightGCN, which proves the effectiveness of the pre-training module. Without RAM-GNN in the pre-training step, our model is similar to LightGCN. The pre-training module incorporates the relations between users and items into the recommendation model so that it can be an effective supplement to the collaborative filtering model.

C. Pre-training Performance (RQ2)

To measure the difference of RAM-GNN with other multi-relational GNN models, we compare our model with five state-of-the-art GNN models. In this section, we only substitute the RAM-GNN in the pre-training step with other models but still use the GNN model in the recommendation step. For the models which cannot learn the embeddings of the item features, we only use the item embeddings in the recommendation step.

1) Compared Methods: We compare the performance of our model with the following five baselines:

- **GCN** [36]: This is the widely-used spectral GCN model. Since it is not designed for multi-relational graphs, we only pick one type of relation for each graph in the pre-training.

[3]https://wsdm-cup-2018.kkbox.events

[4]https://pytorch.org/

TABLE III

EXPERIMENTS ON TWO DATASETS COMPARING OUR PROPOSED RAM-GNN FRAMEWORK WITH EIGHT BASELINE MODELS USING THE METRICS: HIT RATE (HR), MEAN RECIPROCAL RANK (MRR), AND NORMALIZED DISCOUNTED CUMULATIVE GAIN (NDCG). THE BOLD AND UNDERLINED NUMBERS INDICATE THE BEST AND SECOND-BEST RESULTS ON EACH DATASET AND METRIC, RESPECTIVELY. "IMPROVEMENT" MEANS THE MINIMUM IMPROVEMENT AMONG ALL BASELINES.

Datasets	MovieLens						KKBox					
Models	HR@10	MRR@10	NDCG@10	HR@20	MRR@20	NDCG@20	HR@10	MRR@10	NDCG@10	HR@20	MRR@20	NDCG@20
MF-BPR	0.125	0.042	0.061	0.205	0.046	0.079	0.663	0.404	0.465	0.763	0.411	0.489
FM	0.143	0.051	0.072	0.208	0.055	0.087	0.701	0.426	0.493	0.799	0.432	0.515
NFM	0.146	0.054	0.077	0.213	0.057	0.089	0.717	0.441	0.512	0.795	0.443	0.523
FISM	0.132	0.049	0.068	0.209	0.053	0.087	0.696	0.410	0.484	0.764	0.426	0.524
CKE	0.140	0.048	0.069	0.209	0.053	0.088	0.693	0.433	0.495	0.787	0.439	0.531
LightGCN	0.151	0.056	0.079	0.224	0.058	0.094	0.745	0.539	0.568	0.821	0.510	0.546
KGPolicy	0.154	0.058	0.083	0.231	0.062	0.104	0.778	0.562	0.591	0.847	0.554	0.613
RCF	0.159	0.059	0.082	0.235	0.064	0.102	0.794	0.572	0.625	0.856	0.576	0.641
RAM-GNN	**0.164**	**0.062**	**0.087**	**0.242**	**0.066**	**0.107**	**0.812**	**0.584**	**0.647**	**0.873**	**0.589**	**0.660**
Improvement	4.46%	5.08%	4.82%	2.98%	3.13%	2.88%	2.26%	2.10%	3.52%	1.99%	2.08%	2.96%

TABLE IV

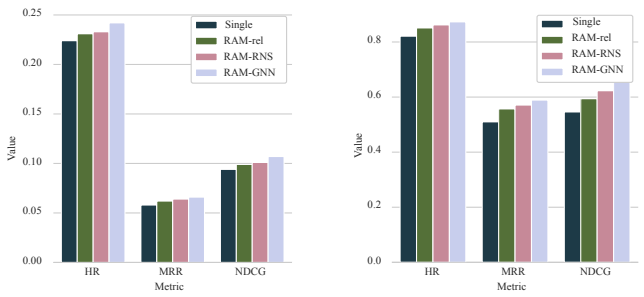
EXPERIMENTS COMPARING OUR PROPOSED RAM-GNN MODEL WITH FIVE BASELINE GNN MODELS IN THE PRE-TRAINING STEP.

Datasets	MovieLens						KKBox					
Models	HR@10	MRR@10	NDCG@10	HR@20	MRR@20	NDCG@20	HR@10	MRR@10	NDCG@10	HR@20	MRR@20	NDCG@20
GCN	0.148	0.049	0.070	0.218	0.051	0.079	0.727	0.516	0.543	0.803	0.498	0.522
R-GCN	0.155	0.056	0.074	0.224	0.055	0.084	0.735	0.522	0.551	0.810	0.501	0.534
W-GCN	0.153	0.057	0.076	0.220	0.057	0.086	0.730	0.523	0.546	0.807	0.505	0.531
VR-GCN	0.151	0.050	0.072	0.226	0.059	0.084	0.734	0.519	0.550	0.802	0.502	0.557
CompGCN	0.155	0.058	0.080	0.233	0.064	0.101	0.759	0.543	0.612	0.832	0.553	0.624
RAM-GNN	**0.164**	**0.062**	**0.087**	**0.242**	**0.066**	**0.107**	**0.812**	**0.584**	**0.647**	**0.873**	**0.589**	**0.660**
Improvement	5.84%	6.90%	8.97%	3.28%	3.13%	5.94%	7.03%	7.48%	5.72%	4.93%	6.51%	5.71%

- **R-GCN** [17]: Relational GCN enhances the GCN by employing different weight matrices to process multiple types of relations.
- **W-GCN** [37]: Weighted GCN utilizes an additional trainable relational-specific scalar weight in the GCN model.
- **VR-GCN** [38]: Vectorized Relational GCN (VR-GCN) learns the embeddings of relations in the GCN framework. Each relation is represented as a vector.
- **CompGCN** [18]: Composition-based R-GCN (CompGCN) employs KG algorithms in GCN models. The relations are initialized as vectors, and the KG algorithm (TransE) is applied to learn the embeddings of nodes and relations.

2) Results: The results comparing pre-training models are shown in Table IV. From the table, we note the following observations:

- RAM-GNN outperforms all baseline GNN models in the experiments. This proves the effectiveness of our proposed RAM-GNN. Also, CompGCN outperforms all other baselines on both datasets, indicating that KG algorithms contribute to learning the embeddings.
- Our model has a greater improvement on the KKbox dataset than on the MovieLens. This indicates that RAM-GNN has a better capability in handling large graphs. We infer the reason can be the attention layers in RAM-GNN. When the graph becomes larger and denser, the attention mechanism can help to understand the importance of each node and relation, which is proven useful when dealing with complex data [39].



(a) MovieLens (b) KKBox

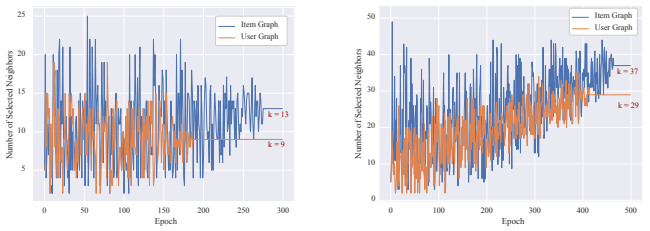
Fig. 4. Ablation Study on datasets MovieLens and KKBox w.r.t the metrics HR@20, MRR@20, and NDCG@20.

D. Ablation Study (RQ3)

In this section, we measure the effectiveness of the pre-training, as well as the node and relation-level attention layers. We compare our model with four ablation models: 1) *Single*, which does not the pre-training step. 2) *RAM-RNS*, which only has the reinforced neighbor sampler in the pre-training model. 3) *RAM-rel*, which only has the relation-level attention layer during pre-training, and 4) *RAM-GNN*, which is the complete model we propose. Figure 4 displays the comparison between the ablation models.

From Figure 4, we have the following observations:

- Both RAM-RNS and RAM-rel outperform the single model, proving the effectiveness of the node-level and relation-level attention layers. In the best performing



(a) MovieLens (b) KKBox

Fig. 5. The changes of filtering threshold in RL process. We choose user ages and item genres as examples for both datasets. The initial values of both datasets are 5, and the change of each iteration is 2

RAM-GNN, there can be multiple relations between two entities. If we treat all types of relations equally in the model, it will miss the difference in the meanings of relations. Besides, the node-level attention layer specifies different weights for different nodes in the neighborhood to improve the aggregation quality. Also, the better performance of RAM-GNN compared to a single model shows the importance of pre-training.

- On the KKBox dataset, the performance of RAM-RNS is clearly better than RAM-rel, while their performance is much closer on MovieLens. This phenomenon may result from the complexity of the user and item graph. When the graph is denser, there can be more nodes connected to the central node. With RNS, the model can emphasize the most similar nodes and reduce redundant information.

E. Filtering Threshold Analysis (RQ4)

In this section, we discuss the filtering threshold learned in the RNS module. The experimental results are shown in Figure 5. We choose user ages and item genres as examples for both datasets. The initial threshold is set as 5, and ϵ is 2.

From the results in Figure 5, we observe that the filtering thresholds on MovieLens dataset converges when $k = 9$ and $k = 13$ for the user and item graph, while on KKBox they converges when $k = 29$ and $k = 37$. Because KKBox has more users and items compared to MovieLens, a larger filtering threshold can ensure the GNN gets enough information in learning the node embeddings.

F. Hyperparameter Analysis (RQ5)

In this section, we conduct experiments on two kinds of hyperparameters: composition operations and embedding dimensions. We compare different hyperparameters and summarize the best settings for our model.

1) Composition Operations: We compare the three composition operations mentioned in Section 2.2.1. The three operations are addition (add), multiplication (mul), and circular-correlation (corr). Based on the RAM-GNN framework, the experimental results are listed in Table V.

From the results, we observe that *corr* has the best performance overall. Therefore we utilize *corr* when conducting the experiments in the previous sections. However, the results on these three operations are similar. The largest difference between operations in Table V is only 3.03%. Therefore, *add* and *mul* are also worth trying during practical use because these operations are simpler to compute, and they can also achieve good performance.

TABLE V
EXPERIMENTS ON THREE COMPOSITION RELATIONS. 'IMPROV' IS THE ABBREVIATION OF 'IMPROVEMENTS'

Datasets	Metrics	Operations			Improv.
		add	mul	corr	
MovieLens	HR@20	0.240	0.237	**0.242**	2.07%
	MRR@20	0.063	**0.068**	0.066	3.03%
	NDCG@20	0.104	0.102	**0.107**	2.88%
KKBox	HR@20	0.863	0.866	**0.873**	0.80%
	MRR@20	0.579	0.583	**0.589**	1.03%
	NDCG@20	0.657	**0.662**	0.660	1.21%

TABLE VI
EXPERIMENTS VARYING THE EMBEDDING DIMENSION ON TWO DATASETS.

Datasets	Metrics	Dimensions				
		20	40	60	80	100
MovieLens	HR@20	0.2383	0.2395	**0.2423**	0.2406	0.2414
	MRR@20	0.0647	0.0658	0.0651	**0.0663**	0.0655
	NDCG@20	0.1052	0.1043	**0.1071**	0.1065	0.1058
KKBox	HR@20	0.859	0.861	0.865	**0.873**	0.863
	MRR@20	0.571	0.575	0.583	0.581	**0.589**
	NDCG@20	0.651	0.655	0.657	**0.661**	0.659

2) Embedding Dimensions: We vary the embedding dimensions from 20 to 100 to see the tendency of the change in predictive performance. The experimental results are displayed in Table VI.

From the results, we observe that the optimal embedding dimension is about 60-80. Also, the optimal embedding dimension on KKbox is larger than on MovieLens. This may because the KKbox dataset is much larger than the MovieLens dataset. When training the model, the larger dataset has a stronger ability to optimize embeddings with higher dimensions.

V. RELATED WORKS

A. GNN-based Recommendation Models

GNNs have proven to be useful in different areas [23], [40]–[45]. There also exists a rich literature utilizing the graph structures in data to provide recommendations. Among them, there are two main directions about which graph to use. One direction is to use the user-item bipartite graph to derive recommendations. Among them, [46]–[48] directly perform convolution operations to explore interactions between users and items. [29], [34] leverage layer-to-layer aggregation functions to capture the high-order connections. [49] models user-item interactions with a dynamic graph. These methods apply GNN on the user-item interaction graph from different aspects. The GNN structure has advantage of representing high-dimensional graph data into low-dimensional embeddings without feature engineering, so it is suitable to be directly

implemented on the user-item interaction graph. However, the interaction graph ignores the rich features of users and items.

The other direction is to use the KG to provide recommendations. This direction contains two different categories of approaches as well. One category is non-GNN models. Many early studies [1], [35], [50]–[52] derive embeddings from KGs via optimization methods and utilize the embeddings in downstream collaborative filtering/prediction steps. These methods are more efficient than recent GNN-based approaches, since they only use the KG embeddings as auxiliary information. However, they tend to be less effective because they fail to incorporate KG information into the end-to-end architecture. The other category is GNN-based models, which tend to apply GNN on KG to derive representations for all nodes (and even relations) [8]–[10], [53]. Besides, [2], [54], [55] construct a heterogeneous graph containing users, items and baskets as three kinds of nodes. These approaches integrate information from both the collaborative bipartite graph and the user and item features, and thus achieve better performance. However, dealing with different relations is still challenging for these models, since GNNs are naturally designed for homogeneous graphs.

B. Multi-Relational Graph Neural Networks

Currently, Graph Neural Networks (GNNs) have been widely explored to process graph-structured data. Motivated by convolutional neural networks, Bruna et al. [56] propose graph convolutions in the spectral domain. Then, Kipf and Welling [36] simplified the previous graph convolution operation and designed a Graph Convolutional Network (GCN) model. To inductively generate node embeddings, Hamilton et al. proposed the GraphSAGE [57] model to learn node embeddings with sampling and aggregation functions. All these models have demonstrated their superior performance on many tasks, e.g., link prediction and node classification.

When the graph has multiple kinds of relations [58] between a pair of nodes, it forms a more complex graph structure, and we call it multi-relational graph. There are some traditional methods handling multi-relational graph including TransE, TransR, DistMult. TransE [16] embeds entities and relations following the translational principle. TransR [59] extends TransE by separating different spaces for entities and spaces. DistMult [19] represents relations as diagonal matrices. These methods have competitive performance on different tasks, but have limitation in recovering missing facts in the multi-relational graph.

After the appearance of GNNs, many studies extended them to deal with multi-relational graph. Relational GCN [17] is the first work that learns node embeddings from multi-relational graphs. For each kind of relation, it utilizes a weight matrix to represent the relation. It demonstrates that R-GCN is able to handing the missing fact issue. Based on this work, weighted GCN [37] adds a trainable weight on each relation in the GCN model. Vectorized Relational GCN (VR-GCN) [38] learns node embeddings as well as relations embeddings with the knowledge base algorithm TransE [16]. Composition-based Relational GCN (CompGCN) [18] is more flexible because it can apply knowledge base algorithms when learning node and relation embeddings. All the models above have a common limitation in that they cannot deal with the graphs that have several relations between two nodes.

VI. Conclusion

In this paper, we introduce a model, Reinforced Attentive Multi-relational Graph Convolutional Network (RAM-GNN), to pre-train the user graph and item graph. RAM-GNN has two main modules. First is relation-level attention which calculates the importance of different types of relations. The second one is Reinforced Neighbor Sampler (RNS), which searches the most similar neighbors iteratively to reduce redundant information and improve efficiency. From the pre-training step, we learn the user and item embeddings. With another GNN for the user-item interaction graph, we inject the knowledge learned from pre-training to the collaborative filtering model and then make recommendations. Experimental results show that our proposed RAM-GNN also has the best performance, as compared to other multi-relational GNNs.

On top of the relations we present in this work, there are some other potential relations among users and items to be explored in the pre-training step, e.g. time dependency of interactions and social relations in users. In the future, we will try to extend our model on sequential recommendation and social recommendation to utilize more kinds of relations and improve performance.

VII. Acknowledgement

The authors of this paper were supported by the S&T Program of Hebei through grant 21340301D, in part by NSF under grants III-1763325, III-1909323, III-2106758, and SaTC-1930941.

References

[1] B. Hu, C. Shi, W. X. Zhao, and P. S. Yu, "Leveraging meta-path based context for top-n recommendation with a neural co-attention model," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1531–1540.

[2] Z. Liu, M. Wan, S. Guo, K. Achan, and P. S. Yu, "Basconv: Aggregating heterogeneous interactions for basket recommendation with graph convolutional neural network," in *Proceedings of the 2020 SIAM International Conference on Data Mining*. SIAM, 2020, pp. 64–72.

[3] J. Tang, X. Hu, and H. Liu, "Social recommendation: a review," *Social Network Analysis and Mining*, vol. 3, no. 4, pp. 1113–1133, 2013.

[4] L. Yang, Z. Liu, Y. Dou, J. Ma, and P. S. Yu, "Consisrec: Enhancing gnn for social recommendation viaconsistent neighbor aggregation." ACM, 2021.

[5] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 353–362.

[6] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "Kgat: Knowledge graph attention network for recommendation," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 950–958.

[7] Y. Lu, Y. Fang, and C. Shi, "Meta-learning on heterogeneous information networks for cold-start recommendation," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1563–1573.

[8] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo, "Ripplenet: Propagating user preferences on the knowledge graph for recommender systems," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 417–426.

[9] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo, "Knowledge graph convolutional networks for recommender systems," in *The world wide web conference*, 2019, pp. 3307–3313.

[10] J. Zhao, Z. Zhou, Z. Guan, W. Zhao, W. Ning, G. Qiu, and X. He, "Intentgc: a scalable graph convolution framework fusing heterogeneous information for recommendation," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2347–2357.

[11] X. Wang, Y. Xu, X. He, Y. Cao, M. Wang, and T.-S. Chua, "Reinforced negative sampling over knowledge graph for recommendation," in *Proceedings of The Web Conference 2020*, 2020, pp. 99–109.

[12] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, pp. 3111–3119, 2013.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019.

[15] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *Advances in neural information processing systems*, 2019, pp. 5753–5763.

[16] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013, pp. 2787–2795.

[17] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*. Springer, 2018, pp. 593–607.

[18] S. Vashishth, S. Sanyal, V. Nitin, and P. Talukdar, "Composition-based multi-relational graph convolutional networks," *arXiv preprint arXiv:1911.03082*, 2019.

[19] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," *arXiv preprint arXiv:1412.6575*, 2014.

[20] M. Nickel, L. Rosasco, and T. Poggio, "Holographic embeddings of knowledge graphs," *arXiv preprint arXiv:1510.04935*, 2015.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, pp. 5998–6008, 2017.

[22] B. Hao, J. Zhang, H. Yin, C. Li, and H. Chen, "Pre-training graph neural networks for cold-start users and items representation," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021, pp. 265–273.

[23] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 315–324.

[24] K.-H. Lai, D. Zha, K. Zhou, and X. Hu, "Policy-gnn: Aggregation optimization for graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 461–471.

[25] H. Peng, R. Zhang, Y. Dou, R. Yang, J. Zhang, and P. S. Yu, "Reinforced neighborhood selection guided multi-relational graph neural networks," *arXiv preprint arXiv:2104.07886*, 2021.

[26] O. Barkan and N. Koenigstein, "Item2vec: neural item embedding for collaborative filtering," in *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2016, pp. 1–6.

[27] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2017, pp. 355–364.

[28] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *European conference on machine learning*. Springer, 2005, pp. 437–448.

[29] 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.

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

[31] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[32] S. Rendle, "Factorization machines," in *2010 IEEE International Conference on Data Mining*. IEEE, 2010, pp. 995–1000.

[33] S. Kabbur, X. Ning, and G. Karypis, "Fism: factored item similarity models for top-n recommender systems," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 659–667.

[34] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 639–648.

[35] X. Xin, X. He, Y. Zhang, Y. Zhang, and J. Jose, "Relational collaborative filtering: Modeling multiple item relations for recommendation," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 125–134.

[36] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *ICLR*, 2016.

[37] C. Shang, Y. Tang, J. Huang, J. Bi, X. He, and B. Zhou, "End-to-end structure-aware convolutional networks for knowledge base completion," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3060–3067.

[38] R. Ye, X. Li, Y. Fang, H. Zang, and M. Wang, "A vectorized relational graph convolutional network for multi-relational network alignment." in *IJCAI*, 2019, pp. 4135–4141.

[39] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[40] H. Peng, J. Li, Q. Gong, Y. Song, Y. Ning, K. Lai, and P. S. Yu, "Fine-grained event categorization with heterogeneous graph convolutional networks," *arXiv preprint arXiv:1906.04580*, 2019.

[41] Z. Liu, X. Li, Z. You, T. Yang, W. Fan, and P. Yu, "Medical triage chatbot diagnosis improvement via multi-relational hyperbolic graph neural network," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 1965–1969.

[42] Z. Liu, X. Li, H. Peng, L. He, and S. Y. Philip, "Heterogeneous similarity graph neural network on electronic health records," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 1196–1205.

[43] Y. Cao, H. Peng, J. Wu, Y. Dou, J. Li, and P. S. Yu, "Knowledge-preserving incremental social event detection via heterogeneous gnns," in *Proceedings of the Web Conference 2021*, 2021, pp. 3383–3395.

[44] J. Li, H. Peng, Y. Cao, Y. Dou, H. Zhang, P. Yu, and L. He, "Higher-order attribute-enhancing heterogeneous graph neural networks," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[45] Y. Hei, R. Yang, H. Peng, L. Wang, X. Xu, J. Liu, H. Liu, J. Xu, and L. Sun, "Hawk: Rapid android malware detection through heterogeneous graph attention networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[46] L. Zheng, C.-T. Lu, F. Jiang, J. Zhang, and P. S. Yu, "Spectral collaborative filtering," in *Proceedings of the 12th ACM Conference on Recommender Systems*, 2018, pp. 311–319.

[47] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.

[48] W. Yu and Z. Qin, "Graph convolutional network for recommendation with low-pass collaborative filters," in *International Conference on Machine Learning*. PMLR, 2020, pp. 10 936–10 945.

[49] X. Li, M. Zhang, S. Wu, Z. Liu, L. Wang, and P. S. Yu, "Dynamic graph collaborative filtering," *arXiv preprint arXiv:2101.02844*, 2021.

[50] C. Luo, W. Pang, Z. Wang, and C. Lin, "Hete-cf: Social-based collaborative filtering recommendation using heterogeneous relations," in *2014 IEEE International Conference on Data Mining*. IEEE, 2014, pp. 917–922.

[51] Y. Xian, Z. Fu, S. Muthukrishnan, G. De Melo, and Y. Zhang, "Reinforcement knowledge graph reasoning for explainable recommendation," in *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, 2019, pp. 285–294.

[52] H. Wang, F. Zhang, X. Xie, and M. Guo, "Dkn: Deep knowledge-aware network for news recommendation," in *Proceedings of the 2018 world wide web conference*, 2018, pp. 1835–1844.

[53] X. Tang, T. Wang, H. Yang, and H. Song, "Akupm: Attention-enhanced knowledge-aware user preference model for recommendation," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1891–1899.

[54] Z. Liu, X. Li, Z. Fan, S. Guo, K. Achan, and S. Y. Philip, "Basket recommendation with multi-intent translation graph neural network," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 728–737.

[55] J. Chang, C. Gao, X. He, D. Jin, and Y. Li, "Bundle recommendation with graph convolutional networks," in *Proceedings of the 43rd international ACM SIGIR conference on Research and development in Information Retrieval*, 2020, pp. 1673–1676.

[56] J. B. Estrach, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and deep locally connected networks on graphs," in *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

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

[58] Y. Wang, Z. Liu, Z. Fan, L. Sun, and P. S. Yu, "Dskreg: Differentiable sampling on knowledge graph for recommendation with relational gnn," *arXiv preprint arXiv:2108.11883*, 2021.

[59] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.