Reinforced Meta-path Selection for Recommendation on Heterogeneous Information Networks

Wentao Ning
The University of Hong Kong
Southern University of Science and Technology
wtning@cs.hku.hk

Reynold Cheng
The University of Hong Kong
ckcheng@cs.hku.hk

Jiajun Shen
TCL Research Hong Kong
sjj@tcl.com

Nur Al Hasan Haldar
The University of Western Australia
nur.haldar@uwa.edu.au

Ben Kao
The University of Hong Kong
kao@cs.hku.hk

Nan Huo
The University of Hong Kong
huonan@connect.hku.hk

Wai Kit Lam
TCL Research Hong Kong
kitkit8120@gmail.com

Tian Li
TCL Research Hong Kong
tian23.li@tcl.com

Bo Tang
Southern University of Science and Technology
tangb3@sustech.edu.cn

Abstract

Heterogeneous Information Networks (HINs) capture complex relations among entities of various kinds and have been used extensively to improve the effectiveness of various data mining tasks, such as in recommender systems. Many existing HIN-based recommendation algorithms utilize hand-crafted meta-paths to extract semantic information from the networks. These algorithms rely on extensive domain knowledge with which the best set of meta-paths can be selected. For applications where the HINs are highly complex with numerous node and link types, the approach of hand-crafting a meta-path set is too tedious and error-prone. To tackle this problem, we propose the *Reinforcement learning-based Meta-path Selection* (RMS) framework to select effective meta-paths and to incorporate them into existing meta-path-based recommenders. To identify high-quality meta-paths, RMS trains a reinforcement learning (RL) based policy network (agent), which gets rewards from the performance on the downstream recommendation tasks. We design a HIN-based recommendation model, HRec, that effectively uses the meta-path information. We further integrate HRec with RMS and derive our recommendation solution, RMS-HRec, that automatically utilizes the effective meta-paths. Experiments on real datasets show that our algorithm can significantly

improve the performance of recommendation models by capturing important meta-paths automatically.

1 Introduction

Nowadays, Heterogeneous Information Networks (HINs) [16] are being extensively used to improve the performance of many novel data mining tasks such as recommender systems [27, 35, 37], natural language processing [34, 46], community detection [7, 31]. A HIN consists of multiple types of entities and links, and provides an advanced data structure that can convey diversity in an intuitive manner. It reveals the complex inter-dependency of each entity in a single graph and provides much side information for recommendation tasks. In Figure 1, the toy example of a movie knowledge-graph with four diverse node types and three types of relations represents a heterogeneous information network.

A real world network containing both the structural and semantic information (e.g., social networks, knowledge graphs, world wide web, etc.), can be modeled as a HIN. Different from the conventional network definition, a HIN can effectively combine various structural information and can contain rich semantics, which benefits real-world applications, such as recommendation systems. For example, traditional recommendation algorithms [20, 24] usually use user-item
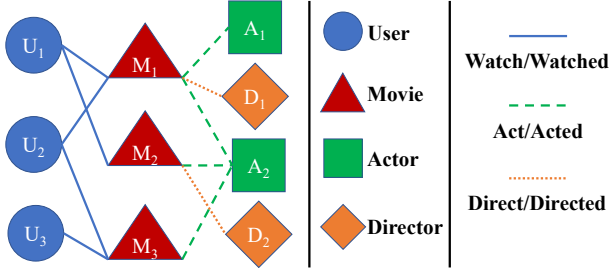
Figure 1. An example of HIN

interaction history to generate results. However, HIN can analyze more information than the interaction logs, which produces more accurate results. Therefore, it received a lot of attention and has been applied in many recommender systems in recent years.

Due to the complex relations on HINs, the concept of *meta-path* [28] is proposed to extract useful information from it. A meta-path consists of a set of node types, which are linked by a set of edge types. In Figure 1, $User \rightarrow Movie \rightarrow Actor \rightarrow Movie$ is a meta-path, which can be adopted to find movies that have the same actor as movies watched by specific users. With the popularity of meta-path, plenty of recommendation algorithms [27, 41] based on meta-paths are proposed. These methods leverage semantic information in different meta-paths to enhance the representation of users and items in the HIN. Some studies [9, 12] employ an attention mechanism to aggregate the information available in different meta-paths.

Though the meta-paths are considered effective for recommendation, however, selecting the most effective meta-path set has not been well addressed in the existing studies mentioned above. The performance of recommendation is highly sensitive to the choice of meta-paths. According to our experiments, using different meta-path sets can result in performance that differs by more than 30%. Current meta-path-based algorithms [8, 12, 40] suppose that meta-paths are chosen by domain experts. However, the difficulties are 1) only a few people have prior knowledge on the design of meta-path, 2) the searching space of the meta-path combination is extremely huge. It is impossible to use brute force to select the optimal meta-path set. 3) the selection of meta-paths is highly dependent on datasets and models. The optimal meta-path set cannot be easily extended to other datasets or algorithms. Han et. al. proposed GEMS [10] that uses a genetic algorithm to search useful meta-structures. However, the enormous searching space of meta-structures results in the inefficiency of this method and this method also cannot be applied in existing methods.

To undertake these challenges, we propose Reinforced Meta-path Selection (RMS), an RL-based method, to figure out meaningful meta-paths for recommendation. We model this problem as a Markov Decision Process (MDP) and use Deep Q-Network [23] (DQN), which is an extensively used reinforcement learning framework, to tackle this complex problem. We train a policy network (agent) to automatically

generate a relation to extend current meta-paths to take the place of the human labor of manual design of meta-path set. The current state is based on the existing meta-paths we selected and the reward function considers the performance gain on the recommendation (NDCG@10), which drives the agent to generate better relations to extend the current meta-path set. However, not all the generated meta-paths are useful and informative, so we also define some strategies to guarantee the generated meta-paths are effective for recommendation.

We also integrate RMS and propose a meta-path-based recommendation algorithm RMS-HRec, without the need to manually specify the meta-paths. For the core recommendation algorithm part, we borrow the idea from HAN [40] and propose a new recommendation model HRec. HRec needs a set of user meta-paths, which start and end with a user type node, and a set of item meta-paths, which start and end with an item type node. It will apply a node-level attention layer to all meta-path neighbors for each meta-path and fuse the information from them by a meta-path-level attention layer. After we get the embeddings of users and items, we multiply them to get the recommendation scores of users to items.

The main contributions of our work are summarized as follows:

- We propose a general meta-path selection framework RMS, which can be plugged into meta-path-based recommendation models. To the best of our knowledge, this is the first framework that has been experimentally proven to be useful for existing meta-path-based recommenders.
- Equipped with RMS, we develop a new meta-path-based recommendation method RMS-HRec and design some strategies during training to fully explore the potential of meta-paths for recommendation tasks.
- We conduct extensive experiments to evaluate the performance of RMS and RMS-HRec. Results demonstrate the effectiveness of both models and reveal our method can identify important meta-paths that may be neglected by manual design.

2 Preliminary

Definition 2.1. (**Heterogeneous Information Network**) A Heterogeneous Information Network (HIN) is a graph, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{N}, \mathcal{R})$, with multiple types of nodes and edges. $\mathcal{V}$ represents the node set, which is associated with a node type mapping function $\phi : \mathcal{V} \rightarrow \mathcal{N}$, where $\mathcal{N}$ is the node type set. $\mathcal{E}$ denotes the edge set, which is also associated with an edge type mapping function $\psi : \mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{R}$ denotes the relation type set.

Example: Figure 1 is an example of HIN, which consists of four types of nodes (i.e., $\mathcal{N}$ = {user, movie, actor, director}) and three types of relations / edges (i.e., $\mathcal{R}$ = {watch/watched, act/acted, direct/directed}).
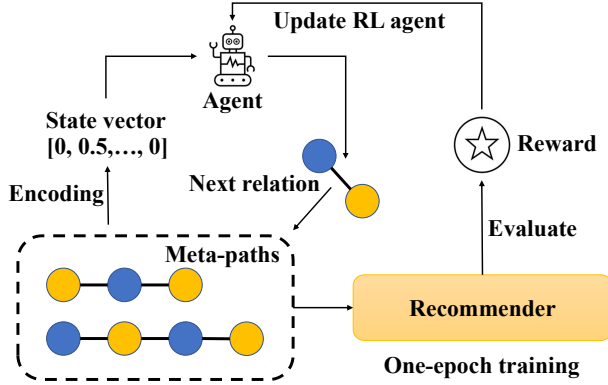
Figure 2. An overview of RMS. The agent is a policy network that aims to investigate high-quality meta-paths. The agent will embed the current meta-path set into a state vector, which is the input of this agent, and it will generate the next relation, which will update the current meta-path set and produce a new set.

Definition 2.2. (**Meta-path**) Given an HIN $\mathcal{G}$, a meta-path $\mathcal{M}$ is a sequence of node types which are linked by relation types in the form of $n_1 \xrightarrow{r_1} n_2 \xrightarrow{r_2} \cdots \xrightarrow{r_l} n_{l+1}$ (abbreviated as $n_1 n_2 \cdots n_{l+1}$), where $n_j \in \mathcal{N}$ denotes the node type and $r_j \in \mathcal{R}$ refers the relation type.

Example: $User \xrightarrow{Watch} Movie \xrightarrow{Acted} Actor \xrightarrow{Act} Movie$ (abbreviated as $UMAM$) is a meta-path in Figure 1.

Definition 2.3. (**Meta-path Instances**) Given an HIN $\mathcal{G}$ and a meta-path $\mathcal{M}$, a meta-path instance $I_i^{\mathcal{M}}$ is defined as a node sequence in $\mathcal{G}$ following the meta-path $\mathcal{M}$.

Example: As shown in Figure 1, suppose meta-path $\mathcal{M} = UMAM$, then one meta-path instance $I_i^{\mathcal{M}}$ can be $U_1 \xrightarrow{Watch} M_1 \xrightarrow{Acted} A_2 \xrightarrow{Act} M_3$.

Definition 2.4. (**Meta-path Neighbors**) Given an HIN $\mathcal{G}$ and a meta-path $\mathcal{M}$, the meta-path neighbors of node n are a set of nodes $N_n^{\mathcal{M}}$ in $\mathcal{G}$ which is connected to n via the meta-path $\mathcal{M}$.

Example: As shown in Figure 1, suppose meta-path $\mathcal{M} = MAM$, then node M_1's meta-path neighbors $N_{M_1}^{\mathcal{M}} = \{M_1, M_3\}$.

3 Methodology

Figure 2 shows the overview of our meta-path selection framework RMS, which uses an agent to generate new meta-path sets and gets rewards from the recommendation model to update the agent. For the recommender, it can be any meta-path-based model, and we also propose a model HRec that adopts HAN [40] to embed users and items. We design some training techniques to train HRec better. Integrated with RMS, we propose a new algorithm RMS-HRec, which

can avoid manually designing meta-paths. The RL agent and the recommender will be co-trained in RMS-HRec.

In this section, we will first introduce our recommender HRec, which takes advantage of the semantic information of meta-paths for recommendation. Then we will elaborate on how the RL agent works, which is the core part of RMS. Last, we will explain how our framework RMS can be adopted in other meta-path-based methods.

3.1 Meta-path-based Recommender

Figure 3 shows the architecture of our recommender. Our recommendation model HRec is a hierarchical model which adopt HAN [40] to embed nodes. It contains a two-level attention mechanism to embed nodes. The first level is node-level attention and the second is meta-path-level attention. Furthermore, the model is trained by a contrastive learning way and enhanced by some strategies we design.

Two-level Attention. Since there are many types of nodes on a HIN, different types of nodes may have different feature spaces. It applies a type-specific projection to project the embeddings of different types of nodes into the same embedding space. After that, the information of each node will be propagated to their meta-path neighbors by a node-level attention layer, then the embedding of each node will be updated. Since different meta-paths may have different importance for a recommendation task, it uses a meta-path-level attention mechanism to fuse the node embeddings from different meta-paths. Since the two-level attention model is not our main contribution, we put the details in Appendix A.

Training strategies. Here, we also propose two strategies which can improve the efficiency and effectiveness of recommendation.

1. In the real world, the graph is often scale-free, which may contain billions of nodes and edges. Also, for some nodes on a HIN, they may have too many meta-path neighbors, which will result in a huge time and memory cost. In most cases, it will encounter an out-of-memory error. Therefore, we employ the Neighbor Sampling (NS) technique, which will randomly sample a subset of meta-path neighbors to perform message passing every time. This technique allows us to avoid running out of memory and save a lot of time.

2. Meta-path should be an effective tool to select semantic-related nodes. However, if a node is connected to too many nodes in the graph via a meta-path, then this meta-path should be ineffective because it cannot distinguish which nodes are highly related to a particular node. Therefore, for each meta-path, we will calculate the average number of the meta-path neighbors of each node. If the average number divided by the total number of nodes is larger than a given threshold t ($0 < t < 1$), we will discard this meta-path and not use it in the recommender.
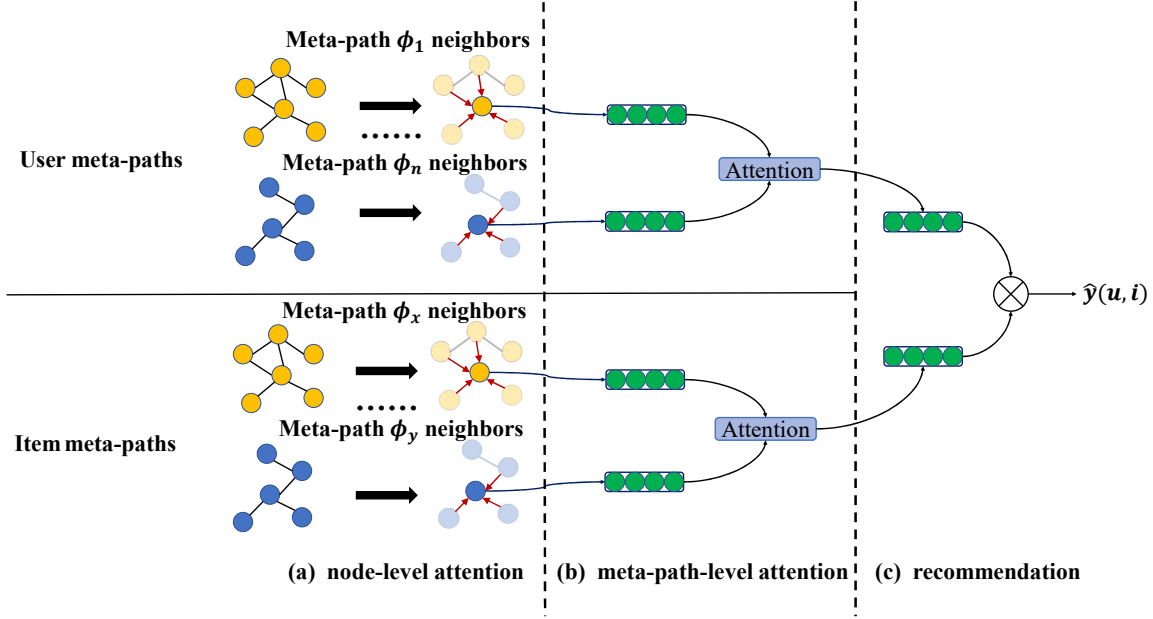
Figure 3. An overview of HRec. The node-level attention is to learn to differentiate the importance of different meta-path neighbors and aggregate the weighted information from them to update the node embeddings. Also, since different meta-paths may play a different role in a recommendation task, the meta-path-level attention is proposed to learn to assign the weight for different meta-paths and fuse the embeddings for each meta-path to get a better representation of nodes. After that, we perform inner product to calculate the recommendation scores and apply a contrastive learning way to train the model.

Recommendation. After we get the embeddings for users and items, we will calculate the inner product of them to predict the scores that how much an item matches a user:

$$\hat{y}(u, i) = h_u^T h_i \qquad (1)$$

To train our recommendation model, we adopt a contrastive learning way and use BPR loss [25], which makes the scores of observed user-item iterations larger than the scores of unobserved interactions:

$$\mathcal{L}_{rec} = \sum_{(u,i,j) \in O} -ln \ \sigma(\hat{y}(u, i) - \hat{y}(u, j)) \qquad (2)$$

where $O = \{(u, i, j) \mid (u, i) \in \mathcal{R}^+, (u, j) \in \mathcal{R}^-\}$ denotes the training set. $\mathcal{R}^+$ means the positive (interacted) user-item pairs and $\mathcal{R}^-$ denotes the negative (un-interacted) user-item pairs, $\sigma(\cdot)$ represents the sigmoid function.

For the training steps, we will adopt the embeddings learned by Matrix Factorization (MF) [25] as the initial embeddings of user and item nodes. Then we optimize the loss $\mathcal{L}_{rec}$ using Adam [18], a widely used optimizer that dynamically controls the learning rate.

3.2 RL-based Meta-path Selection

To get user/item embeddings in HRec, we need meta-paths that start and end with a user/item type node to make sure that the meta-path neighbors are users/items. We propose an RL-based framework RMS to select these meta-paths for the

recommender. This framework can not only be used in our recommendation model but also exiting meta-path-based algorithms. Here, we elaborate on the procedure of RMS.

Reinforcement learning follows a Markov Decision Process (MDP) [30] formulation, which contains a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a decision policy $\mathcal{P}$ and a reward function $\mathcal{R}$. The agent learns to take actions based on the current state in the environment derived from a HIN to maximize the cumulative rewards. We design the key components of RMS and formalize it with the quartuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$. Since our recommendation model needs both meta-paths which start and end with users and items, so we train two RL agents to generate user meta-paths and item meta-paths separately. The procedure is formulated as below:

- **State ($\mathcal{S}$):** To guarantee that the agent can take an optimal action based on the current state, the state needs to store the structural information of current meta-paths. Here, for each meta-path, we use the relation IDs to represent it. To encode a meta-path, we will use a binary vector and assign high bits (1) to the positions which are contained in that meta-path, then assign low bits (0) to other positions. For instance, if there are 6 kinds of relations on a HIN[1], we will assign the relation IDs to 1...6. if the ID representation of a

---

[1]Note that relations with opposite head and tail nodes should be seen as two different relations, e.g. A-M and M-A should be two different relations.

meta-path ϕ is $[2, 6, 4]$, then its encoding E_ϕ will be $(0, 1, 0, 1, 0, 1)$.

The state s_i at step i is represented by the embedding of the meta-path set Φ_i at step i. To encode a meta-path set, we just add up all of the vector representations of the meta-path in that meta-path set and apply a L2 normalization as following:

$$s_i = Normalize(\sum_{\phi \in \Phi_i} E_\phi) \qquad (3)$$

where Φ_i denotes the meta-path set at step i, E_ϕ represents the encoding of meta-path ϕ.

Note that we use $\{User - Item - User\}$ as the initial user meta-path set and $\{Item - User - Item\}$ as the initial item meta-path set for the initial step. The initial user/item meta-path set contains only one meta-path that starts and ends with a user/item and reflects the user-item interactions, which is crucial for recommendation tasks.

- **Action ($\mathcal{A}$):** The action space $\mathcal{A}_{s_i}$ for a state s_i is all the relations $(r_1, r_2, ...)$ that appear on a HIN, with a special action $STOP(r_0)$. At each step, the policy network will predict an action to extend the current meta-path set to yield a higher reward during the long period. Here, if the policy network selects the action $STOP$ or the current step exceeds the maximal step limit I, the meta-paths will not be extended.

  If the predicted action is a relation in the HIN, we concatenate it with a complementary relation to make it a symmetric meta-path. This is done to ensure that the generated meta-paths are also symmetric and start and end with a user/item. Then we try to extend all the current meta-paths with this relation after auto-completion and also try to insert this relation into the current meta-path set. Note that the previous meta-paths will also be copied and preserved in the meta-path set and will not be removed.

  For instance, if the current user meta-path set is $\{U - M - U\}$ and the predicted action (relation) is $M - A$, then we will auto-complete it with the relation $A - M$ and it will become $M - A - M$. The meta-path set at the next step will be $\{U - M - U, U - M - A - M - U\}$. However, $M - A - M$ does not start with a user type, so it will not be added to the user meta-path set.

- **Policy ($\mathcal{P}$):** The decision policy $\mathcal{P}$ is a state transition function that searches for the optimal action based on the current state. The goal is to train a policy that can try to maximize the discounted cumulative reward $R = \sum_{i=i_0}^{I} \gamma^{i-i_0} r_i$, where I is the maximal step limit and γ is a real number between 0 and 1. γ is acted as a discount factor to make the far future reward less important than near future reward.

In our problem, states are in a continuous space and actions are in a discrete space. Therefore, we use a typical RL algorithm, DQN [23], although it can be substituted with other RL algorithms. The basic idea of DQN is a Q-function which is a neural network and can compute a Q-value which is used to evaluate an action on a state: $\mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. Suppose that we have a Q-network Q, we can directly construct a policy π that maximizes the reward:

$$\pi(s) = \underset{a}{argmax}\, Q(s, a) \qquad (4)$$

And for the update of Q, it will obey the Bellman equation [23]:

$$Q(s_i, a_i) = r + \gamma Q\,(s_{i+1}, \pi\,(s_{i+1})) \qquad (5)$$

where s_{i+1} is the next state and a_{i+1} is the next action. In our algorithm, we utilize MLP as the Q-network since it can approximate any function. Note that it can also be replaced by any neural network.

- **Reward ($\mathcal{R}$):** Reward is to evaluate the decision made by the policy network, which helps the agent to get better performance. Here, if the agent choose $STOP$, the reward will be 0, if the agent choose any relations which will not change the meta-path set, to punish this situation, the reward will be -1. Otherwise, we define the reward as the improvement of the performance. The formulation can be represented as follow:

$$R(s_i, a_i) = N(s_i, a_i) - N(s_{i-1}, a_{i-1}) \qquad (6)$$

Here, $N(s_i, a_i)$ is the performance of recommendation task at step i. Here, we adopt the NDCG@10 as the performance metric, but other metrics could also be used. Note that the reward can be less than 0. After the policy network predicts the action, we extend the meta-path set and put them into the recommendation model to perform one-epoch training. To have an efficient training process, we evaluate the model on a randomly selected small test set to get the performance.

Training and optimization. The whole procedure of training can be reviewed in Figure 2: i) get the current state based on the current meta-path set. ii) generate the next action (relation) based on the current state. iii) extend the current meta-path set and get the next state. iv) put the new meta-path set into the recommender and get the reward to update the RL agent. When we train the agent, we first calculate the temporal difference error δ:

$$\delta = Q(s, a) - \left(r + \gamma \max_{a} Q\,(s', a)\right) \qquad (7)$$

In order to minimize this error, we use the Huber loss [17], which is shown as below:

$$\mathcal{L} = \frac{1}{|B|} \sum_{(s,a,s',r) \in B} \mathcal{L}(\delta) \qquad (8)$$

where $\mathcal{L}(\delta) = \begin{cases} \frac{1}{2}\delta^2 & |\delta| \leq 1 \\ |\delta| - \frac{1}{2} & \text{otherwise} \end{cases}$, B is randomly sampled history data.

3.3 Apply in other recommenders

According to the introduction of RMS, it can not only be used in our recommendation model but also in other meta-path-based models. To the best of our knowledge, the meta-path-based solutions can be divided into two categories. The first category needs two kinds of meta-paths, one is the meta-paths that start and end with the user type, and the other is the meta-paths that start and end with the item type. The second category uses the meta-paths which start with the user type and end with the item type.

Our recommendation algorithm belongs to the first category and we also integrate HERec [27], which is also a first-category method, into our RL framework in our experiments. The results show that our RL method greatly improves the performance of recommendation. For the second category, it is also very easy to adapt to RMS. We just need to use $\{User - Item\}$ as the initial meta-path set and the following procedures remain the same. We also equip RMS with MCRec [12], which belongs to the second category, in our experiments. The experiment results show that RMS can also effectively improve its performance.

4 Experiments

We conduct experiments to answer these research questions.

- **RQ1:** How does the RMS-explored meta-path set perform compared with different meta-path sets?
- **RQ2:** How does RMS perform compared to other meta-path selection strategies?
- **RQ3:** How does RMS-HRec perform compared to other state-of-the-art recommendation models?

4.1 Experiment Settings

Datasets. We conduct our experiments on two wildly used datasets Yelp[2] and Douban Movie[3]. The statistics of them are shown in Table 1 and the schema are shown in Figure 4. The detail descriptions of datasets are in Appendix B.1.

Experiments Methods and Metrics. To evaluate the performance of recommendation, we split our dataset into training, validating and testing set with 8:1:1 ratio. We adopt the leave-one-out evaluation. For each user, we will regard the interacted items as positive items and the remaining items as negative items. For each positive item, we randomly select 499 negative items and rank the positive item among 500 items. Then we adopt two common metrics, *Hit Ratio*

[2]https://www.yelp.com/dataset/download
[3]https://movie.douban.com/

Dataset	Entity	Relation
Yelp	#User (U): 16,239	# U - B: 198,397
	#Business (B): 14,284	# U - U: 158,590
	#Compliment (Co): 11	# U - Co: 76,875
	#Category (Ca): 511	# B - Ci: 14,267
	#City (Ci): 47	# B - Ca: 40,009
Douban Movie	#User (U): 13,367	#U - M: 1,068,278
	#Movie (M): 12,677	#U - G: 570,047
	#Group (G): 2,753	#U - U: 4,085
	#Actor (A): 6,311	#M - A: 33,587
	#Director (D): 2,449	#M - D: 11,276
	#Type (T): 38	#M - T: 27,668

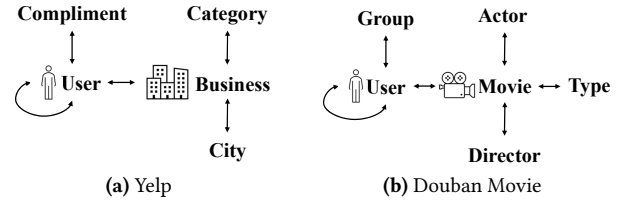**Table 1.** Statistics of the datasets

Figure 4. Graph schema of two datasets

at Rank k (HR@k) and *Normalized Discounted Cumulative Gain at Rank k* (NDCG@k). HR@k indicates whether the test positive item is ranked at top k and NDCG@k assign higher scores if the ranking position of the test item is higher.

Meta-path-based Recommenders. To prove that RMS can be applied in all meta-path-based recommenders. We not only adopt it on RMS-HRec, but also on two state-of-the-art meta-path-based models (HERec [27] and MCRec [12]).

Other Baseline Models. In order to show the recommendation performance of RMS-HRec, we also compare six state-of-the-art models besides the above two models. They contain Matrix Factorization (MF)-based models (BPR [25], NCF [11]), regularization-based models (CKE [44], CFKG [1]) and GNN-based models (GEMS [10], KGAT [37]). The detail descriptions of all the baseline models are in Appendix B.2.

Baseline Meta-path Selection Strategies. Since this is the first work to be applied to existing recommendation algorithms, in order to prove the effectiveness of RMS in meta-path-based recommenders, We also design two baseline meta-path selection strategies and compare them with RMS.

- **Random**: We run several experiments until the time limit is reached. For each experiment, we randomly select a set of meta-paths. The number of meta-paths is a random number between 1 and 4. To ensure fairness, we also do one-epoch training and evaluation. After that, we select the best meta-path set to train and test.
- **Greedy**: We first initialize our meta-path set as the same as RMS and run several experiments until the time limit is reached. For each experiment, we randomly select a set of meta-paths. We also perform one-epoch training and evaluating on the training set

Dataset	Meta-path set	HR1	HR3	NDCG10
Yelp	RMS	**0.0648**	**0.1484**	**0.1740**
	- UBUU	0.0600	0.1441	0.1696
	- BUB	0.0569	0.1352	0.1597
	- BCiB	0.0594	0.1399	0.1657
	- BCaB	0.0599	0.1425	0.1670
	+ UBCiBU	0.0552	0.1340	0.1602
	+ UBCaBU	0.0626	0.1463	0.1721
	+ BUBCaB	0.0564	0.1375	0.1636
	+ BUBCiB	0.0590	0.1439	0.1688
	Init.	0.0486	0.1237	0.1514
Douban Movie	RMS	**0.0997**	**0.2131**	**0.2400**
	- UMU	0.0971	0.2027	0.2308
	- UMDMU	0.0968	0.2075	0.2345
	- MUM	0.0984	0.2091	0.2360
	- MAM	0.0970	0.2127	0.2381
	- MDM	0.0948	0.2076	0.2357
	- MUMDM	0.0962	0.2083	0.2349
	- MDMAM	0.0979	0.2104	0.2375
	- MAMAM	0.0989	0.2103	0.2384
	+ UMAMU	0.0958	0.2096	0.2358
	+ UMUU	0.0952	0.2064	0.2333
	Init.	0.0798	0.1763	0.2015

Table 2. Effect of meta-path on RMS-HRec

Dataset	Model	Strategy	HR1	HR3	NDCG10
Yelp	HRec	RMS	**0.0648**	**0.1484**	**0.1740**
		Greedy	0.0489	0.1181	0.1449
		Random	0.0589	0.1381	0.1633
	HERec	RMS	**0.0389**	**0.0979**	**0.1215**
		Greedy	0.0374	0.0918	0.1166
		Random	0.0344	0.0922	0.1167
	MCRec	RMS	**0.0548**	**0.1317**	**0.1540**
		Greedy	0.0516	0.1234	0.1504
		Random	0.0526	0.1229	0.1451
Douban Movie	HRec	RMS	**0.0997**	**0.2131**	**0.2400**
		Greedy	0.0882	0.1946	0.2221
		Random	0.0835	0.1800	0.2076
	HERec	RMS	**0.0594**	**0.1613**	**0.1984**
		Greedy	0.0571	0.1577	0.1960
		Random	0.0559	0.1575	0.1945
	MCRec	RMS	**0.0928**	**0.1961**	**0.2236**
		Greedy	0.0918	0.1952	0.2204
		Random	0.0903	0.1928	0.2193

Table 3. Effect of different meta-path selection strategies

for each meta-path, then we expand our meta-path set with the best one. After that, we use the expanded meta-path set to train and test on the test set.

To ensure fairness, we make sure that all of the strategies will run within the same time limit. Note that we only perform one-epoch training and evaluation when we train the RL agent. After the RL agent is well trained on the training set, we use the meta-path set generated by the RL agent and train the recommender until the model converges. Then test the recommender on the whole test set. The implementation details of our experiments are in Appendix B.3.

4.2　Meta-path Sensitivity Analysis (RQ1)

To illustrate the influence of meta-paths on recommendation performance and prove the effectiveness of RMS, we run RMS-HRec on both datasets and it returns {UBU, UBUU, BUB, BCiB, BCaB} on Yelp dataset and {UMU, UMDMU, MAM, MDM, MUM, MUMDM, MAMAM, MDMAM} on Douban Movie dataset. To test the meta-path sensitivity of RMS-HRec, we adopt leave-one-out principle to see the performance change if we remove one meta-path from it. Similarly, we also try to add one meta-path and test it. Finally, we test the performance of HRec using the initial meta-path set to see how much performance improvement RMS brings. The results are reported in Table 2. Here, '-' means remove a meta-path, '+' means add a meta-path and 'Init.' means the initial meta-path set. We have the following findings:

- The meta-path set found by RMS performs better than all of the other meta-paths on both datasets, which

proves that our approach is effective and can figure out the optimal ones.

- Compared with the performance using the initial meta-path, RMS get 33.3%, 20.0% and 14.9% performance improvement in terms of HR@1, HR@3 and NDCG@10 on Yelp dataset. On Douban Movie dataset, RMS gets 24.9%, 20.8% and 19.1% performance gain. This suggests that using meaningful meta-paths can greatly improve the performance of recommendation.

- When we remove **BUB** on Yelp dataset or **MDM** on Douban Movie dataset, the performance drop will be larger than when we remove other meta-paths. This demonstrates that different meta-paths have different impacts on recommendation performance. When we add another meta-path into our meta-path set, the performance also drops. It indicates that using as many meta-paths as possible does not necessarily lead to performance gains. This also proves that finding optimal meta-path set by human labor is very difficult.

4.3　Meta-path Selection Method Study (RQ2)

To demonstrate the effectiveness of RMS, we design two meta-path selection strategies as our baselines. We set the running time limit of all strategies to be the same and test these strategies on both HRec and existing meta-path-based models. We report the experimental results in Table 3 and have some observations and conclusions as following:

- In both datasets and all the algorithms, our proposed RMS constantly outperforms random and greedy strategies on all metrics. This proves that our method can find effective meta-paths for recommendation in a more efficient way than other baselines.

	Yelp					Douban Movie				
	HR1	HR3	HR10	NDCG10	NDCG20	HR1	HR3	HR10	NDCG10	NDCG20
RMS-HRec	**0.0648**	**0.1484**	**0.3213**	**0.1740**	**0.2095**	**0.0997**	**0.2131**	**0.4258**	**0.2400**	**0.2802**
BPR	0.0388	0.1025	0.2592	0.1301	0.1638	0.0529	0.1421	0.3502	0.1768	0.2218
NCF	0.0514	0.1251	0.2927	0.1522	0.1872	0.0622	0.1605	0.3854	0.1974	0.2438
CFKG	0.0456	0.1092	0.2630	0.1360	0.1710	0.0574	0.1495	0.3668	0.1862	0.2323
CKE	0.0572	0.1246	0.2721	0.1477	0.1791	0.0634	0.1646	0.3930	0.2015	0.2479
HERec	0.0389	0.0979	0.2381	0.1215	0.1528	0.0594	0.1613	0.3910	0.1984	0.2424
MCRec	0.0548	0.1317	0.2887	0.1540	0.1876	0.0928	0.1961	0.3985	0.2236	0.2631
GEMS	0.0100	0.0294	0.0868	0.0408	0.0583	0.0270	0.0603	0.1293	0.0742	0.0945
KGAT	0.0415	0.1151	0.2718	0.1388	0.1733	0.0630	0.1644	0.3922	0.2009	0.2469

Table 4. Overall Performance comparison of RMS-HRec

- In HRec algorithm, we find that RMS outperforms random by 32.5%, 25.7% and 20.1% on HR@1, HR@3 and NDCG@10 on Yelp dataset. Also, it has similar results on Douban Movie dataset. This demonstrates that random meta-paths are commonly not useful. Greedy algorithm outperforms random in most instances, but it is still not good as RMS.

- We also find that in some algorithms such as MCRec, the performance improvement is not as large as in HRec. We argue that the performance gain highly depends on the algorithms. Some models do not leverage meta-paths effectively so that their performance is not highly sensitive to the selected meta-paths.

4.4 Recommendation Effectiveness (RQ3)

The main results of recommendation performance comparison are shown in Table 4. Here, we use the best meta-path set found by RMS when we run HERec and MCRec. The major observations are summarized as follows:

- Our method outperforms all the baselines over all metrics on both datasets. This result shows that meta-paths play an important role in recommendation on HINs and our method can effectively leverage the semantic and structural information of the HINs and perform recommendation.

- RMS-HRec outperforms two meta-path-based methods (HERec and MCRec). This demonstrates that our method can maximize the roles of meta-paths in recommendation tasks on HINs. RMS-HRec can better utilize meta-paths to model the entities on HINs.

- RMS-HRec outperforms two MF-based methods (BPR, NCF). This is because that HINs have more entities besides users and items, it contains more semantic information. RMS-HRec also has a better performance compared to two GNN-based methods (GEMS, KGAT), this shows that meta-paths can be a better way to explore information from HINs.

- GEMS gets a bad performance in our experiments, this indicates it has limited power to leverage metastructures to capture the complex collaborative signals.

Furthermore, the search space of GEMS is extremely large and it also leads to inefficiency.

5 Related Work

Recommendation on HINs. HIN-based recommenders can be divided into three categories: 1) *Embedding-based methods.* These algorithms usually extract information from HINs directly to enrich the embeddings of users and items, such as CKE [44] , CFKG [45], KSR [14], DKN [35]. 2) *Path-based methods.* The path-based methods usually exploit the connectivity between users and items on a HIN which contains users, items, and other entities. This kind of algorithms [12, 15, 27, 29, 38] can also improve the explainability since they can find which paths are more important. 3) *Graph Neural Network (GNN)-based methods.* GNN-based solutions [4, 36, 37, 39] leverage both semantic information and connectivity information on a HIN for recommendation. Message passing is applied in these methods to propagate information to the neighbors of each node iteratively to update the embeddings.

Meta-path-based recommenders. Recently, Meta-paths are wildly used in a large volume of HIN-based algorithms. Metapath2vec [5] is a HIN embedding method that adopts a random work strategy on meta-paths for node embedding. However, this method can only use one meta-path that may result in information loss. HAN [40] improves this drawback and utilizes an attention mechanism to fuse embeddings from different meta-paths. PEAGNN [9] generates meta-path subgraphs for each meta-path first, then perform a GNN layer on each sub-graph and fuse embeddings from them via an attention mechanism. MEIRec [6] is an intent recommendation algorithm that will recommend queries to users. They utilize meta-paths to select related neighbors and design a GNN to obtain the embeddings of users and queries. However, all of these methods suppose that meta-paths are already given, which is not realistic in real applications.

Meta-path discovery on HINs. Some work [22, 26, 33] attempts to discover meaningful meta-paths on HINs. FSPG [22] and MPDRL [33] are the methods in data mining fields for meta-path discovery based on several node pairs. However, they require users to provide node pairs and cannot be applied in existing recommenders. GEMS [10] adopts a genetic

algorithm to find effective meta-structures for recommendation and performs GCN [19] layers to fuse information from the meta-structures. However, these meta-structures can only be used in their recommender due to the path format and also very time-consuming to find them because of the huge search space. Some graph embedding methods (e.g. GTN [42], HGT [13]) can weight all meta-paths via attention mechanisms. Nevertheless, they do not give a specific meta-path set, so it cannot be used in existing recommendation solutions. Furthermore, different algorithms may have different requirements in the form of meta-paths. However, the above algorithms cannot find the specified form of meta-paths (e.g. start and end with a user type) so that they cannot be used in existing methods.

6 Conclusion

In this paper, we proposed a reinforcement learning (RL)-based meta-path selection framework RMS to automatically select meaningful meta-paths for meta-path-based recommendation models to play to their maximum ability. We designed the state and reward function to enable the RL agent to capture the features and importance of meta-paths. Also, we proposed a recommendation model and design some training strategies to fully explore the potential of meta-paths. Extensive experiment results demonstrate the meta-paths may significantly influence the performance of recommendation but our method can effectively find the meaningful ones for recommendation.

References

[1] Q. Ai, V. Azizi, X. Chen, and Y. Zhang. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms*, 11(9):137, 2018.

[2] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. *NeurIPS*, 26, 2013.

[3] J. S. Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In *NeuIPS*, pages 211–217, 1990.

[4] C. Chen, M. Zhang, W. Ma, Y. Liu, and S. Ma. Jointly non-sampling learning for knowledge graph enhanced recommendation. In *SIGIR*, pages 189–198. ACM, 2020.

[5] Y. Dong, N. V. Chawla, and A. Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *SIGKDD*, pages 135–144. ACM, 2017.

[6] S. Fan, J. Zhu, X. Han, C. Shi, L. Hu, B. Ma, and Y. Li. Metapath-guided heterogeneous graph neural network for intent recommendation. In *SIGKDD*, pages 2478–2486. ACM, 2019.

[7] Y. Fang, Y. Yang, W. Zhang, X. Lin, and X. Cao. Effective and efficient community search over large heterogeneous information networks. *PVLDB*, 13(6):854–867, 2020.

[8] X. Fu, J. Zhang, Z. Meng, and I. King. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *WWW*, 2020.

[9] Z. Han, M. U. Anwaar, S. Arumugaswamy, T. Weber, T. Qiu, H. Shen, Y. Liu, and M. Kleinsteuber. Metapath- and entity-aware graph neural network for recommendation. *CoRR*, abs/2010.11793, 2020.

[10] Z. Han, F. Xu, J. Shi, Y. Shang, H. Ma, P. Hui, and Y. Li. Genetic meta-structure search for recommendation on heterogeneous information network. In *CIKM*, page 455–464, 2020.

[11] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *WWW*, pages 173–182, 2017.

[12] 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 *SIGKDD*, pages 1531–1540, 2018.

[13] Z. Hu, Y. Dong, K. Wang, and Y. Sun. Heterogeneous graph transformer. In *WWW*, pages 2704–2710. ACM / IW3C2, 2020.

[14] J. Huang, W. X. Zhao, H. Dou, J. Wen, and E. Y. Chang. Improving sequential recommendation with knowledge-enhanced memory networks. In *SIGIR*, pages 505–514. ACM, 2018.

[15] X. Huang, Q. Fang, S. Qian, J. Sang, Y. Li, and C. Xu. Explainable interaction-driven user modeling over knowledge graph for sequential recommendation. In *MM*, pages 548–556. ACM, 2019.

[16] Z. Huang, Y. Zheng, R. Cheng, Y. Sun, N. Mamoulis, and X. Li. Meta structure: Computing relevance in large heterogeneous information networks. In *SIGKDD*, pages 1595–1604, 2016.

[17] P. J. Huber. Robust Estimation of a Location Parameter. *Ann. Math. Stat.*, 35(1):73 – 101, 1964.

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[19] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*. OpenReview.net, 2017.

[20] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[21] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, 2015.

[22] C. Meng, R. Cheng, S. Maniu, P. Senellart, and W. Zhang. Discovering meta-paths in large heterogeneous information networks. In *WWW*, pages 754–764. ACM, 2015.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[24] S. Rendle. Factorization machines. In *ICDM*, pages 995–1000, 2010.

[25] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[26] B. Shi and T. Weninger. Mining interesting meta-paths from complex heterogeneous information networks. In *ICDM*, pages 488–495. IEEE Computer Society, 2014.

[27] C. Shi, B. Hu, W. X. Zhao, and P. S. Yu. Heterogeneous information network embedding for recommendation. *TKDE*, 2018.

[28] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB*, 4(11):992–1003, 2011.

[29] Z. Sun, J. Yang, J. Zhang, A. Bozzon, L. Huang, and C. Xu. Recurrent knowledge graph embedding for effective recommendation. In *RecSys*, pages 297–305. ACM, 2018.

[30] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. *IEEE Trans. Neural Networks*, 9(5):1054–1054, 1998.

[31] C. Tu, X. Zeng, H. Wang, Z. Zhang, Z. Liu, M. Sun, B. Zhang, and L. Lin. A unified framework for community detection and network representation learning. *TKDE*, 31(6):1051–1065, 2019.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NeuIPS*, pages 5998–6008, 2017.

[33] G. Wan, B. Du, S. Pan, and G. Haffari. Reinforcement learning based meta-path discovery in large-scale heterogeneous information networks. In *AAAI*, pages 6094–6101. AAAI Press, 2020.

[34] D. Wang, P. Liu, Y. Zheng, X. Qiu, and X.-J. Huang. Heterogeneous graph neural networks for extractive document summarization. In

ACL, pages 6209–6219, 2020.

[35] H. Wang, F. Zhang, X. Xie, and M. Guo. DKN: deep knowledge-aware network for news recommendation. In *WWW*, pages 1835–1844, 2018.

[36] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, and Z. Wang. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *SIGKDD*, pages 968–977. ACM, 2019.

[37] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua. Kgat: Knowledge graph attention network for recommendation. In *SIGKDD*, 2019.

[38] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, and T. Chua. Explainable reasoning over knowledge graphs for recommendation. In *AAAI*, pages 5329–5336. AAAI Press, 2019.

[39] L. Xia, C. Huang, Y. Xu, P. Dai, X. Zhang, H. Yang, J. Pei, and L. Bo. Knowledge-enhanced hierarchical graph transformer network for multi-behavior recommendation. In *AAAI*, pages 4486–4493. AAAI Press, 2021.

[40] W. Xiao, J. Houye, S. Chuan, W. Bai, C. Peng, Y. P., and Y. Yanfang. Heterogeneous graph attention network. *WWW*, 2019.

[41] X. Yu, X. Ren, Y. Sun, B. Sturt, U. Khandelwal, Q. Gu, B. Norick, and J. Han. Recommendation in heterogeneous information networks with implicit user feedback. In *RecSys*, pages 347–350, 2013.

[42] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim. Graph transformer networks. In *NeurIPS*, pages 11960–11970, 2019.

[43] D. Zha, K. Lai, Y. Cao, S. Huang, R. Wei, J. Guo, and X. Hu. Rlcard: A toolkit for reinforcement learning in card games. *CoRR*, abs/1910.04376, 2019.

[44] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W. Ma. Collaborative knowledge base embedding for recommender systems. In *SIGKDD*, pages 353–362. ACM, 2016.

[45] Y. Zhang, Q. Ai, X. Chen, and P. Wang. Learning over knowledge-base embeddings for recommendation. *CoRR*, abs/1803.06540, 2018.

[46] J. Zheng, F. Cai, Y. Ling, and H. Chen. Heterogeneous graph neural networks to predict what happen next. In *COLING*, 2020.

A Two-level Attention Mechanism

Figure 3 shows the architecture of two-level attention mechanism and we describe the details here.

A.1 Node-level Attention

Note that each meta-path neighbor of a node may have different importance, so we will learn the weight of each meta-path neighbor and aggregate them to form node embeddings before we fuse the information from different meta-paths. Since there are many types of nodes in a HIN, different types of nodes may have different feature spaces. We apply a type-specific projection to project the embeddings of different types of nodes into the same embedding space. The procedure can be formulated as follow:

$$z_i = W_\phi x_i \qquad (9)$$

where W_ϕ is the projection matrix of node type ϕ, x_i and z_i are the original and projected embedding of node i respectively.

After that, suppose nodes i and j are connected via a meta-path ϕ, the attention score e_{ij}^ϕ can be calculated using the projected embeddings of nodes i and j by:

$$e_{ij}^\phi = \sigma(\vec{a}_\phi^T[z_i|z_j]) \qquad (10)$$

where z_i and z_j are the projected embeddings of nodes i and j by Equation 9, $\vec{a}_\phi$ is the node-level attention vector for meta-path ϕ, e_{ij}^ϕ is the attention score of nodes i and j for meta-path ϕ, σ denotes the activation function. Note that the attention score of node i to node j may be different to the attention score of node j to node i, which means they may influence each other differently.

Then for a node i, we calculate all the attention score e_{ij} for node $j \in N_i^\phi$ by Equation 10, where N_i^ϕ represents the meta-path neighbors of node i for meta-path ϕ. After that, we get the importance coefficient α_{ij} by normalizing the attention scores using Softmax [3]:

$$\alpha_{ij}^\phi = \frac{\exp(e_{ij}^\phi)}{\sum_{k \in N_i^\phi} \exp(e_{ik}^\phi)} \qquad (11)$$

Next, we will aggregate neighbor embeddings weighted by the attention scores via the following equation:

$$h_i^\phi = \sigma\left(\sum_{j \in N_i^\phi} \alpha_{ij}^\phi z_j\right) \qquad (12)$$

Here, h_i^ϕ is the learned representation of node i for meta-path ϕ and σ is the activation function. Notice that this attention mechanism can also be extended to multi-head attention [32].

A.2 Meta-path-level Attention

Suppose that we have X meta-paths $\{\phi_1, \ldots, \phi_X\}$, after node-level attention, we get X groups of node embeddings, denoted as $\{H_1, \ldots, H_X\}$. Since different meta-paths may have different importance for a recommendation task, we propose a meta-path-level attention mechanism to fuse the node embeddings from different meta-paths.

To get the importance of each meta-path, we first transform the embeddings via a multi-layer perceptron (MLP), then multiply with a meta-path-level attention vector $\vec{q}_\phi^T$. Then get the average of all the meta-path-specific node embeddings. The equation is shown as following:

$$w^{\phi_x} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \vec{q}_{\phi_x}^T \cdot \sigma(MLP(h_i^{\phi_x})), \ h_i^{\phi_x} \in H_x \qquad (13)$$

Here, σ is the activation function and $\mathcal{V}$ is the set of all meta-path-specific nodes.

After that, we use softmax function to normalize the meta-path importance as following:

$$\beta^{\phi_x} = \frac{\exp(w^{\phi_x})}{\sum_{x=1}^{X} \exp(w^{\phi_x})} \qquad (14)$$

Here, β^{ϕ_x} denotes the normalized importance of meta-path ϕ_x. Lastly, we aggregate the embeddings from each

meta-path to get the final node embeddings H according to their weights by:

$$H = \sum_{x=1}^{X} \beta^{\phi_x} \cdot H_x \tag{15}$$

B Experiments

B.1 Datasets

We use two publicly available datasets to conduct our experiments.

- **Yelp**[4] is a subset of Yelp's businesses, reviews, and user data, containing 16,239 users, 14,284 businesses, 11 compliments, 511 categories and 47 cities. It is used for business recommendation and contains over 0.4 million relations.
- **Douban Movie**[5] is a movie recommendation dataset. It contains the attributes and social relations of users and items. It consists of 13,367 users, 12.677 movies, 2,753 groups, 6,311 actors, 2,449 directors and 38 types and has over 1.6 million relations.

B.2 Baseline models

Meta-path-based Recommenders

- **HERec** [27]: HERec generates node sequences for graph embedding by a meta-path based random walk strategy. Then the learned node embeddings are integrated into an extended matrix factorization (MF) model. HERec needs two kinds of meta-path sets. One is the meta-paths start and end with a user type node and the other is the meta-paths start and end with an item type node.
- **MCRec** [12]: MCRec adopts a neural co-attention mechanism which leverages meta-paths context for recommendation. MCRec needs only one kind of meta-path set. The meta-paths should start with a user type node and end with an item type node.

Other baseline models

- **BPR** [25]: This is the traditional matrix factorization model which utilizes Bayesian personalized ranking.
- **NCF** [11]: A matrix factorization model combined with deep neural networks, which is used to learn the latent features of users and items.
- **CKE** [44]: A regularization-based model that using TransR [21] to extract items' structural information.
- **CFKG** [1]: This model utilizes TransE [2] in a united user-entity graph and regards recommendation as a link prediction task.
- **GEMS** [10]: This method propose a genetic algorithm to search meta-structures and utilizes GCN to train the embeddings for users and items.

- **KGAT** [37]: A GNN-based methods which adopt graph neural networks and message passing in a united user-entity graph.

B.3 Implementation details

We implement all the models using Pytorch and DGL[6]. Parameters are randomly initialized and the models are optimized by Adam [18]. For the recommendation part of RMS-HRec, We fix the embedding size to 64, the hidden layer size to 32, batch size to 90000. We set the threshold to 0.5, dropout ratio to 0.1 and learning rate to 0.01. For the DQN of RMS, we adopt the implementation in [43] and conduct a 3-layer MLP with (32, 64, 32) hidden units for Q function. We set the learning rate to 0.001 and the memory buffer size to be 10000. We set the maximal step limit to 4 for RMS-HRec, 3 for HERec and 5 for MCRec.

---

[4]https://www.yelp.com/dataset/download
[5]https://movie.douban.com/

[6]http://dgl.ai/