Meta-Path Discovery Based on Temporal Equivariant Graph Representations

Chen Cheng
Department of Computer Science
Shanghaitech University
chengchen@shanghaitech.edu.cn

Weizhen Zhou
Department of Computer Science
Shanghaitech University
zhouwzh@shanghaitech.edu.cn

Abstract

With the advent of large-scale heterogeneous information, heterogeneous information networks(HIN) is widespread in various fields. Recently, meta-path based recommenders(MPRs) becomes a prevailing method to extract relation from HIN, but most MPRs ignore the dynamic evolution of the nodes and edges attributes which is very important for its expressivity. To incorporate temporal information, we use the time-then-graph to generate temporal graph representation. Meanwhile, to discover the effective meta-path for MPRs, we purpose a Reinforcement Learning based meta-path selection framework work that optimized by a Deep Q-Network.

1 Introduction

Heterogeneous information networks (HINs) J.Huang.et.al [17], Z.Huang.et.al [44], H.Wang.et.al [15], F.Zhang.et.al [10] are logical networks consisting of multiple types of objects and multiple types of links (relations) between them, and Fig1 illustrates an example of paper HINs. The use of HINs is widespread in fields such as natural language processing D.Wang.et.al [8], J.Zheng.et.al [18], social media networks, and Wikipedia's knowledge network. In recent years, reasoning with paths over user-item knowledge graphs (KGs) has become a popular way to build explainable recommenders Y.Xian.et.al [40], X.Wang.et.al [37], K.Zhao.et.al [20]. And due to HIN's strong ability to provide more information for recommenders, they've been receiving plenty of attention of late Q.Ai.et.al [26], Y.Dong.et.al [38], H.Ji.et.al [13].

The challenge is that there are many types of nodes and links in heterogeneous networks, which makes conventional network embedding techniques impractical. Y.Dong.et.al [38]

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

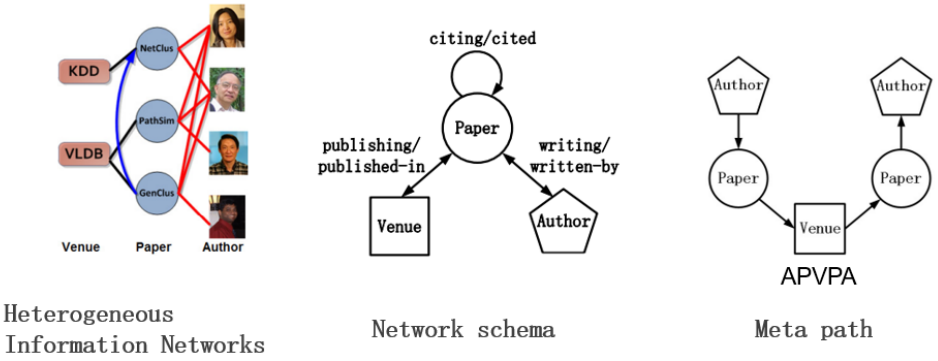

Figure. 1. On the far left is a Heterogeneous Information Network that shows the relation between venue, paper and author. By analysing its network schema we can get its corresponding meta-path which is the abstract relationship between node types and edge types.

The method to solve this is utilizing meta-path Y.Sun.et.al [39], L.Tang.et.al [22], which consists of relations among different node types, used for recommending HINs based on their information. In Figure 1 (from Shi and Philip [31]), by graph' s network schema, we can redefine HIN and construct meta-path. For example, Author → Paper → Venue → Paper → Author(abbreviated as APVPA) is a meta-path and can be adopted to find authors having paper on the same venue.

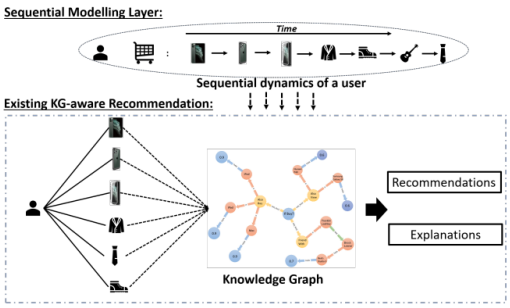

Figure. 2. Existing Sequential modelling and static graph based modelling.

However, most meta-path-based recommenders (MPRs) ignore the dynamic and evolving nature of user-item interactions in real-world recommendation scenarios when analyzing HINs for explainable recommendations Chen et al. [3]. Meanwhile, both recommendation precision and explanations of a user's true intent depend strongly on the dynamics and evolution of users' interactions with items Chen et al. [3], Fang et al. [9].An example of this is the case in Figure 2 (Chen et al. [3]), where someone purchased a phone case and a phone screen protector film after purchasing a brand new phone. If we ignore the sequential information between each purchase (shown in Figure 2," sequential modelling layer") and treat the whole information as a static graph to form our existing KG-aware recommendation, the system probably gives an explanation of buying the phone case is because a similar customer also bought this phone case by exploring co-purchasing relationships. Though the explanation might be valid and the underlying reasoning may help with recommendation, it is still a sub-optimal approach. As a result, it is critical

that a system can generate possible explanations that consider timing and sequential interactions between the user and the item.

In this paper we use the time-then-graph node representation formalized by J.Gao.et.al [16], to incorporate temporal information and get a static equivariant graph representations. In this framework, nodes and edges are represented sequentially as they evolve over time. Then, it uses these temporal representations to define a static graph representation. It has been theoretically proven that time-then-graph architectures that use GNNs with expressivity limited to 1-WL power Xu.et.al [35], Morris.et.al [24] as building blocks have an expressivity advantage over time-and-graph architectures that also use the same 1-WL GNN architecture as building blocks J.Gao.et.al [16].

Another major challenge for MPRs except temporal representations is how to select the meta-path Meng et al. [23]. Existing MPRs require a set of meta-paths as input Hu et al. [14], Meng et al. [23]. And experiments reveal that the performance of MPRs is highly sensitive to the choice of meta-paths Hu et al. [14], Meng et al. [23]. While most existing MPR works claim that domain experts' knowledge can be used to obtain meta-paths, this is not efficient. Because selecting correct meta-path for recommendation is very challenging due to the high time complexity and fact that the structures of the meta-path required by different MPRs and vary S.Fan.et.al [30], Z.Han.et.al [43], B.Hu.et.al [1], C.Shi.et.al [6].

In this paper we use a model-independent Reinforcement learning (RL)-based Meta-path Selection (RMS) framework to select the meaningful meta-paths for recommendation. First, we train a policy network (agent) to automatically generate a relation to extend the current meta-path set to take the place of the human intervention in the designing of meta-path sets and use the encoding of the current meta-path set as input. Second, we update the meta-path selection policy network by exploiting the performance gain of the newly generated meta-path set from the downstream recommender. Finally, we follow a Deep Q-Network (DQN) algorithm to optimize the model.

2 Related Work

2.1 Temporal information

Metapath-based recommender (MPR) (aggregation of meta-path instances)(B.Hu.et.al [1]) is a prevailing method for extracting information from graph structures. Nevertheless, the dynamic nature of real-world user-item interactions is often overlooked by most MPRs which makes it less expressive. In this section we will take an **overview** of the existing work on dealing with temporal information of graph structures and provide an **analysis** and **criticism** of them.

2.1.1 Sequential Information

To capture sequential information and deal with temporal graph, a path-based explainable recommendation problem has been studied in some previous works to some extent. X.Wang.et.al [36] uses LSTM to model the sequential dynamics, but it only consider the sequential information within specific path and ignore that the users' historical sequences is of great importance for it reflects users' dynamic interactions and preferences with items. To improve this, Q.Zhu.et.al [27] combines the path connectivity between users and items and user's historical action and for recommendation. However, it only considers the relation information of user-item paths and ignore the important relation information between item-item, which may restrict the expressiveness of users' temporal evolving information on recommendation.

2.1.2 Temporal Graph representation

On the other hand, some works have focused on GNNs and predicting node attribute evolution in temporal graphs from the perspective of learning equivariant representations.

Time-and-graph: Time-and-graph is the most common representation adopted in the iterature (Li.et.al [21], Chen.et.al [4], Seo.et.al [29]). Where equivariant graph like GNN and sequence representations like RNN are used together to represent the node attributes' temporal evolution in the graph. And in this way, methods of temporal graph neural networks (TGNN)(Karpoor.et.al [19]) produce graph representations for each graph snapshot and embed them over time to generate the final temporal graph representation.

Time-then-graph In recent years, some works have proposed methods that are different from time-and-graph for approaching temporal graphs. TGAT, proposed by da.Xu.et.al [7] utilizes all edges from past snapshots to constructs a static HIN that extracts representations from the constructed graph as the final TGNN. And TGN, proposed by Rossi.et.al [28] extends TGAT, and uses sequence representations of all edges connecting to the same nodes to replace node attributes in the heterogeneous graph.

Based on the above research, an alternative representation framework for temporal graph have be proposed by J.Gao.et.al [16] which denoted as time-then-graph. Where the temporal evolution of node and edge attributes will be sequentially represented firstly. Then it utilizes these temporal representations to define a static graph representation. Compared to time-and-graph, time-then-graph provides several advantages in learning the temporal dynamics of each user. (i) Time-then-graph with 1-WL GNNs is

more expressive than time-and-graph with 1-WL GNNs. (ii)Time-then-graph is computationally more efficient in practice. A task provided in Figure 3 (J.Gao.et.al [16])where any time-and-graph representation will fail while a time-then-graph would work which prove that the time-then-graph is more expressive than time-and-graph.

In this paper, we use the time-then-graph node representation (J.Gao.et.al [16]) to model temporal dependencies with item-item meta-path instances. And the generated final temporal graph representation is used as input to MPRs.

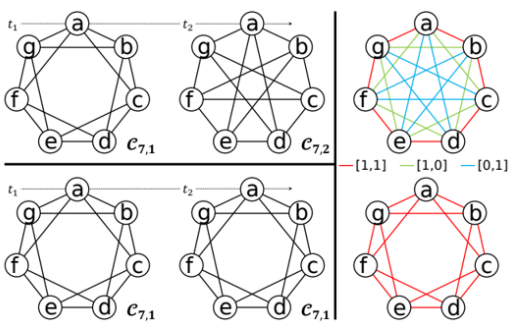

Figure. 3. A synthetic task that shows time-then-graph is more expressive. The top and bottom 2 temporal graph has snapshots at time t_1, t_2,denoted as $C_{7,1}, C_{7,2}$, and two aggregated temporal graphs on the right side. Where 1,0 denote the existing of edge attributes. Time-and-graph will fail to distinguish structure difference between top and bottom since GNN always output the same node representation. But time-then-graph will work since the aggregated forms are different.

2.2 Meta-Path Selection

Existing MPRs require a set of meta-path as input and experiments reveal that the performance of MPRs is highly sensitive to the choice of meta-paths. While most existing MPR works claim that relative expert knowledge can be used to obtain meta-paths which is not efficient. In this section, we will take an **overview** of the existing work on metapath selection and provide an **analysis** and **criticism** of them.

2.2.1 GEMS

GEMS proposed in Z.Han.et.al [42] generates meta-path structures using a genetic algorithm, which are abstract graphs with node types and edge types. Then performs GCN(T.N.Kipf.et.al [33]) layers to combine information from the newly generated meta-path structure. Nevertheless, since different MPRs have different requirement on the meta-path set, these method can only be used in their recommender. Also, it could be highly expensive because of the complexity of finding m length-l meta-paths for n node types is exponential.

2.2.2 RL-HGNN

RL-HGNN proposed in Z.Zhong.et.al [45] utilizes a reinforcement learning based method to find a effective meta-path for each node to learn its representations. However, they focus on producing powerful node embeddings based on found meta-paths, instead

of format-specific embeddings. And then they improve the GNN model meta-paths to support meta-path-based recommendation algorithms. Also, because all of their RL components are designed for obtaining better node embeddings, it is difficult to adapt RL-HGNN to other scenarios and not easy to be adapted to generate meta-paths with some specific forms.

2.2.3 RSM

Based on the above work, we use a RL-based Meta-path Selection (RMS) framework to select meta-path for recommendtion and optimize the selection policy network by a Deep Q-Network (DQN) algorithm. Next we will discuss how to use a RL-based framework RMS to select meta-paths for the MPRs. How do we formulate the selection process as an MDP to formalize the model. And how to optimize the model by a deep Q-learning Network algorithm.

RMS framework uses an agent (policy network) to generate new meta-path set and predict the action (relation in current HIN) to extend the current meta-path set. The recommender takes the newly generated meta-path as input and its performance reward is used to update the agent.

The main steps of training are: (i) get the current state based on the current meta-path set; (ii) generate the next action (relation) by agent and current knowledge;(iii) extend the current meta-path set by newly generated relation; (iv) the recommender uses the new meta-path set as input and get the reward to update the RL agent; (v) when RL agent is fully-trained, let RL agent generate relations from the initial state and extend the meta-path set until the STOP action or until the current step exceeds the maximal step limit I.

Main Component of RSM Reinforcement Learning (RL) follows a Markov Decision Process (MDP)Sutton.et.al [32] formulation, which contains a set of states $\mathbf{S}$, a set of actions $\mathbf{A}$, a decision policy $\mathbf{P}$, and a reward function $\mathbf{R}$. To maximize the cumulative rewards, the agent learns to consider actions based on the current state derived from an HIN. The component of RMS is formalized as the tuple $(\mathbf{S}, \mathbf{A}, \mathbf{P}, \mathbf{R})$ and the specific implementation process is shown in Appendix A.

3 Contribution

In this paper, we use time-then-graph to get the final representation of temporal graph. So that we can conduct RSM on a temporal graph to generate its meaningful meta-path and feed it to MPRs for recommender. Also, we use a Deep Q-Network (DQN) to optimize the policy network in RL-based meta-path selection, of which the specific process is illustrated in Figure 4

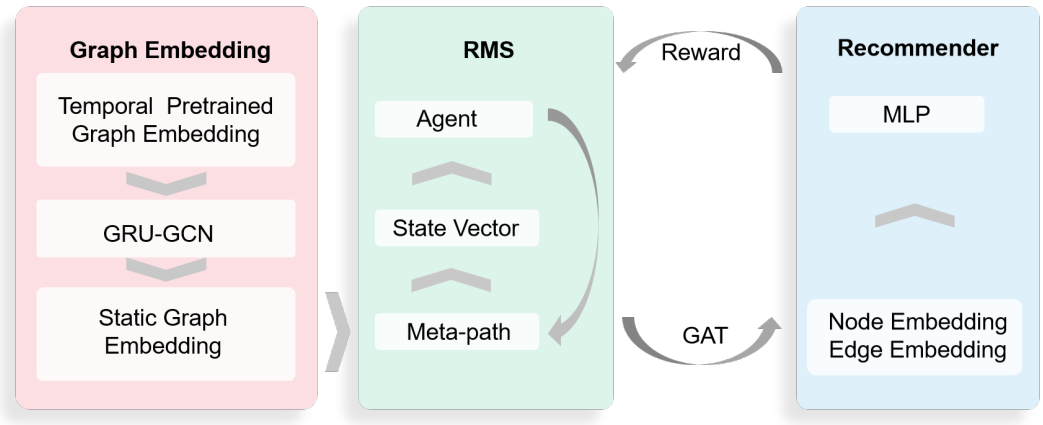

Figure. 4. **Graph Embedding** In this stage, we used two *temporal pretrained graph embedding* dataset. The pretrained embedding are fed into *GRU-GCN* to capture the temporal information between snapshots of the graph. Finally, we get an equivariant graph embedding. **RMS** Based on the embedding constructed in the firts stage, we construct a *markov decision process (MDP)* to construct select *meta-path*. The meta-paths are fist encoded into the *state vectors* and are processed by a *Deep-Q-Networok (DQN)* (i.e. *Agent*). The DQN coordinates with the *recommender* which exerts as a reward function to update the meta-path set. **Recommender** The *recommender* plays the role as an Q function in the DQN and is approximated by an MLP.

We use a typical RL algorithm, Deep Q-Network V.Mnih.et.al [34] (DQN) to optimize this model. Suppose that the policy network denoted as $\mathbf{Q}(s, a)$, which used to evaluate an action on a state. The evaluation process is denoted as: $\mathbf{S} \times \mathbf{A} \rightarrow \mathbb{R}$. We can construct a policy π that maximizes the reward which is represented as:

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

The Bellman equation V.Mnih.et.al [34] is used to update $\mathbf{Q}(s, a)$ as follows:

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

where s_{i+1} is the next state and a_{i+1} is the next action. Specifically, we use multi-layer perceptron (MLP) as the policy network $\mathbf{Q}$ due to its strong ability to approximate function. During the training, we first caculate the temporal difference error δ as below:

$$\delta = Q(s, a) - (r + \gamma \underset{a}{max} Q(s^{'}, a)).$$

Then, in order to minimize the error δ, we define the loss error ε as shown below:

$$\varepsilon = \frac{1}{|V|} \sum_{(s,a,s^{'},r) \in B} \delta^2,$$

where V is randomly sampled history data.

4 Experiment

Corresponding to the two challenges we found, we performed experiments to answer the following two research questions:

- RQ1: How does the DQN-based meta-path selection strategy perform compared to random and greedy selection strategy?
- RQ2: How does *time then graph* representation generated by the *GRU-GCN* model for a temporal graph exerts on the meta-path selection result in RMS?

4.1 Experiment Settings

4.1.1 Datasets.

The HINs our experiment based on is two commonly used public datasets **Douban Movie** [1] and **Yelp** [2]. **Douban Movie** consists of $37,595$ nodes,$3,429,882$ and 12 relations and **Yelp** consists of $31,092$ nodes and $976,276$ edges and 10 relations.

4.1.2 Performance Metrics

Leave-one-out evaluation Han et al. [12] was used in our study. Furthter, *Hit Ratio at Rank K (HR@k) and Normalized Discounted Cumulative Gain at Rank k (NDCG@k)* were used to evaluated the performance.

4.1.3 Implementation Details.

To incorporate the temporal information to nodes embedding and edge embedding, we implement the **GCN-GRU** model following the description of Gao and Ribeiro [11] and its released code using *TSL* Cini and Marisca [5] which is an extension of *Pytorch* Paszke et al. [25]. The *GRU-GCN* model was used to process the pre-trained embeddings of the nodes and edges. To deal with the huge graph data, we use DGL[3] to load the graph data and corporate with the Pytorch-based DQNagent and the recommender which is a 3-layered Multi Layer Perceptron (MLP). For the DQNagent ,we adopt the implementation in Zha et al. [41] using *gym* Brockman et al. [2].

4.2 Meta-path Selection Algorithm Comparison

We compare the *Random*, *Greedy* and *Reinforcement learning* based meta-path selection algorithm. From the top half of the Table 1 we can easily find that DQN outperforms random and greedy on the two datasets and all metrics which means that DQN is

[1] `https://movie.douban.com/`

[2] `https://www.yelp.com/dataset/download`

[3] `https://www.dgl.ai/`

	Yelp				Douban Movie			
Strategy	HR@3	HR@10	NDCG@10	NDCG@20	HR@3	HR@10	NDCG@10	NDCG@20
RL	0.1407	0.3085	0.1647	0.1994	0.2074	0.3753	0.2370	0.2602
Greedy	0.1273	0.2682	0.1543	0.1797	0.1635	0.3144	0.1897	0.2170
Rahdom	0.1076	0.1968	0.1335	0.1466	0.1723	0.3325	0.2005	0.2278
Pretrained Embedding Aggregated by GRU-GCN								
RL	0.1487	0.3230	0.1720	0.2012	0.2163	0.3892	0.2401	0.2714
Greedy	0.1299	0.2732	0.1603	0.1867	0.1700	0.3251	0.2063	0.2260
Rahdom	0.1053	0.1934	0.1293	0.1437	0.1735	0.3376	0.2020	0.2312

Table 1: (1)The upper part of the table shows the effectiveness of the RL algorithm (2)The lower part of the table shows the possible performance improvement for the meta-path selection algorithm.

effective for meta-path selection. By comparing the upper and lower parts of the table, we can find that the embedding preprocessed by the GRU-GCN model which embeds the temporal information can slightly improve the performance of RL and greedy algorithm but seems to have not effect on the random algorithm.

5 Conclusion

In this paper, we implement the time-then-graph method to generate a temporal graph representation. We propose a reinforcement learning-based meta-path selection framework (RMS) that uses Deep Q-Network to determine meaningful meta-path sets for meta-path-based recommendations. This allows the recommender system works on temporal graph, and to achieve a better performance when the meta-path is set for recommendation. Our experiment show that the DQN-based meta-path selection strategy is effective for meta-path selection and outperforms on most of dataset. And time-then-graph model which preprocessed by GRU-GCN model can improve the performance of RL agent in some scenarios but has no significant change on somt specific cases.

References

[1] B.Hu.et.al. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. *SIGKDD*, pages 1531–1540.

[2] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[3] H. Chen, Y. Li, X. Sun, G. Xu, and H. Yin. Temporal meta-path guided explainable recommendation. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 1056–1064, 2021.

[4] Chen.et.al. Graph convolution embedded lstm for dynamic link prediction. *arXiv preprint arXiv:1812.04206*.

[5] A. Cini and I. Marisca. Torch Spatiotemporal, 3 2022. URL `https://github.com/TorchSpatiotemporal/tsl`.

[6] C.Shi.et.al. Heterogeneous information network embedding for recommendation. *TKDE*, 2018.

[7] da.Xu.et.al. Inductive representation learning on temporal graphs. *ICLR 2020 : Eighth International Conference on Learning Representations*.

[8] D.Wang.et.al. Heterogeneous graph neural networks for extractive document summarization. *ACL*, pages 6209–6219, 2016.

[9] H. Fang, D. Zhang, Y. Shu, and G. Guo. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM Transactions on Information Systems (TOIS)*, 39(1):1–42, 2020.

[10] F.Zhang.et.al. Collaborative knowledge base embedding for recommender systems. *SIGKDD*, page 353–362, 2016.

[11] J. Gao and B. Ribeiro. On the equivalence between temporal and static equivariant graph representations. In *International Conference on Machine Learning*, pages 7052–7076. PMLR, 2022.

[12] 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 *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 455–464, 2020.

[13] H.Ji.et.al. Who you would like to share with? a study of share recommendation in social e-commerce. *AAAI Press*, pages 232–239.

[14] 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*, pages 1531–1540, 2018.

[15] H.Wang.et.al. Dkn: deep knowledge-aware network for news recommendation. *WWW*, pages 1835–1844, 2018.

[16] J.Gao.et.al. On the equivalence between temporal and static equivariant graph representations. *International Conference on Machine Learning. PMLR*, pages 7052–7076.

[17] J.Huang.et.al. Improving sequential recommendation with knowledge-enhanced memory networks. *SIGIR*, pages 505–514, 2018.

[18] J.Zheng.et.al. Heterogeneous graph neural networks to predict what happen next. *COLING*, 2020.

[19] Karpoor.et.al. Examining covid-19 forecasting using spatio-temporal graph neural networks. *arXiv preprint arXiv:2007.03113*.

[20] K.Zhao.et.al. Leveraging demonstrations for reinforcement recommendation reasoning over knowledge graphs. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 239–248, 2020.

[21] Li.et.al. Predicting path failure in time-evolving graphs. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery*, pages 1279–1289.

[22] L.Tang.et.al. Relational learning via latent social dimensions. *KDD'09*, pages 817–826.

[23] C. Meng, R. Cheng, S. Maniu, P. Senellart, and W. Zhang. Discovering meta-paths in large heterogeneous information networks. In *Proceedings of the 24th international conference on world wide web*, pages 754–764, 2015.

[24] Morris.et.al. go neural: Higher-order graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence,volume 33*, pages 4602–4609.

[25] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[26] Q.Ai.et.al. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms, 11(9)*, page 137.

[27] Q.Zhu.et.al. A knowledge-aware attentional reasoning network for recommendation. *AAAI*, pages 6999–7006.

[28] Rossi.et.al. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*.

[29] Seo.et.al. Structured sequence modeling with graph convolutional recurrent networks. *International Conference on Neural Information Processing*, pages 362–373.

[30] S.Fan.et.al. Metapath-guided heterogeneous graph neural network for intent recommendation. *SIGKDD*, pages 2478–2486.

[31] C. Shi and S. Y. Philip. *Heterogeneous information network analysis and applications*. Springer, 2017.

[32] R. S. Sutton.et.al. Reinforcement learning: An introduction. *IEEE Trans. Neural Networks, 9(5):*, pages 1054–1054.

[33] T.N.Kipf.et.al. Semi-supervised classification with graph convolutional networks. *ICLR. OpenReview.net*.

[34] V.Mnih.et.al. Human-level control through deep reinforcement learning. *Nature, 518(7540)*, pages 529–533.

[35] Xu.et.al. How powerful are graph neural networks. *International Conference on Learning Representations*.

[36] X.Wang.et.al. Explainable reasoning over knowledge graphs for recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33*, pages 5329–5336.

[37] X.Wang.et.al. Explainable reasoning over knowledge graphs for recommendation. *Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33*, pages 5329–5336, 2019.

[38] Y.Dong.et.al. metapath2vec: Scalable representation learning for heterogeneous networks. *SIGKDD*, pages 135–144.

[39] Y.Sun.et.al. Mining heterogeneous information networks: Principles and methodologies. *Synthesis Lectures on Data Mining and Knowledge Discovery*.

[40] Y.Xian.et.al. Reinforcement knowledge graph reasoning for explainable recommendation. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 285–294, 2019.

[41] D. Zha, K.-H. Lai, Y. Cao, S. Huang, R. Wei, J. Guo, and X. Hu. Rlcard: A toolkit for reinforcement learning in card games. *arXiv preprint arXiv:1910.04376*, 2019.

[42] Z.Han.et.al. Genetic meta-structure search for recommendation on heterogeneous information network. *CIKM*, pages 455–464.

[43] Z.Han.et.al. Metapath- and entity-aware graph neural network for recommendation. *CoRR, abs/2010.11793*, 2020.

[44] Z.Huang.et.al. Meta structure: Computing relevance in large heterogeneous information networks. *SIGKDD*, pages 1595–1604, 2016.

[45] Z.Zhong.et.al. Reinforcement learning enhanced heterogeneous graph neural network. *CoRR, abs/2010.13735*.

A Appendix

Specific implementation process of MDP

State(S): We assign an index to each type of relation in an HIN, and encode each meta-path with a vector of length n (where n is the number of relation types). Each entity of the vector is number of the corresponding relation type in this meta-path. For instance, if there are 4 types of relation in an HIN, and the index representation of a meta-path ϕ is [2,1,1,4], then its encoding E_{phi} will be (2,1,0,1). This will gurantee that all the encodings have same length. The state s_i at step i is represented by the embedding of the meta-path set Φ_i at step i. We apply an L2 normalization to all of the encodings of meta-path in the meta-path set to embed a meta-path set:

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

where Φ_i represents the meta-path set at step i, E_Φ denotes the encoding of meta-path ϕ. By this representation, we can reflect the weights of each relation in a meta-path set.

Action(A): The possible action for state s_i is all the relations that appear on an HIN, denote as $(r_1, r_2, ...)$ with a special action STOP(r_0). At each step, the agent will predict an action to extend the current meta-path set and yeild a new reward. And if the agent predict the action STOP or the current step exceeds the maximum step limit I, the meta-path will not be extended.

During the prediction, we will make sure the extended meta-paths start or end with the same user/item by making it a symmetric meta-path. Then we will extend all the meta-paths in the current meta-path set with this symmetric meta-path and insert this symmetric meta-path into the current meta-path set to generate new meta-path set. Also, the old meta-paths will be copied in the newly meta-path set so that important information will not be neglected.

For instance, if we want a meta-path set that starts and ends with a user type node, in a paper HIN (see Figure 1), the current set is $\{V-P-V\}$ and the predicted relation is $P-A$,then we will complete the relation as $P-A-A$, since $P-A-A$ does not start with a venue type node it will not be added to the meta-path set. So the update meta-path will be $\{V-P-V, V-P-A-P-V\}$. This method makes sure the newly generatedd meta-paths meet the requirements of the recommender.

Policy(P): The policy **P** is a state transition function that finds action based on the current state. **P** intended to train a policy that maximize the cumulative reward which represented as:

$$R = \sum_{i=i_0}^{I} \gamma^{i-i_0} r_i,$$

where I is the maximal step limit and $\gamma \in [0, 1], \gamma \in R$, γ is acted as a discount factor to make the further reward less important than a near reward.

Rewar(R): Reward(**R**) evaluates the decision made by agent and return a real number as reward. Here, if the agent chooses action STOP, the reward will be 0. If the agent chooses any relation which will not change the meta-path set, we need to punish this state and the reward will be -1. Otherwise, we define the reward as the improvement of performances and the reward can be represented as follows:

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

where $N(s_i, a_i)$ is the perfermance of recommendation task at step i. After the policy predict the relation, we extend the current meta-path set and use them to update the agent (policy network).