Heterogeneous Graph Representation for Knowledge Tracing

Jisen Chen, Jian Shen, Ting Long, Liping Shen(✉), Weinan Zhang, and Yong Yu(✉)

Shanghai Jiao Tong University, Shanghai, China
{chenjisen,r_ocky,longting,lpshen,wnzhang}@sjtu.edu.cn
yyu@apex.sjtu.edu.cn

Abstract. Knowledge tracing (KT) is a fundamental task of intelligent education, which traces students' knowledge states by their historical interactions. In KT, students, questions, concepts, and answers are four main types of entities, and they contain various relations, including student-question interactive relations, question-concept relations, and question-answer relations. Such rich knowledge in these heterogeneous relations could potentially improve the prediction of KT. However, it has not been sufficiently utilized in existing KT methods. In this paper, we propose a novel method, called Heterogeneous Graph Representation for Knowledge Tracing, to leverage these useful relations. Our method first models all the complex entities and relations in KT as a heterogeneous graph, and then uses a heterogeneous graph neural network to obtain entities' feature representations. After that, we feed the feature embeddings to a KT model in an end-to-end training manner. Due to the heterogeneous graph's high representational capacity, our method exploits the relations among students, questions, concepts, and answers in a concise and unified way. Experiments on four KT datasets show that our method achieves state-of-the-art performance.

Keywords: Knowledge Tracing · Heterogeneous Graph · Intelligent Education

1 Introduction

Currently, online education is developing rapidly and has gradually become a common way of learning. Many advanced algorithms are used to mine large-scale interactive data between students and the system to provide more intelligent educational services so that each student can have a better adaptive learning experience. *Knowledge tracing* (KT) [7] is a fundamental task of intelligent education, which uses students' historical learning interactive data to trace students' dynamic knowledge states and predict their performances in future interactions.

In KT, the collected historical data include relational information among entities of *students* (i.e., users), *questions* (i.e., problems/exercises), *concepts* (i.e., knowledge concepts/skills), and *answers* (i.e., responses). These entities

M. Tanveer et al. (Eds.): ICONIP 2022, LNCS 13623, pp. 224–235, 2023.
https://doi.org/10.1007/978-3-031-30105-6_19

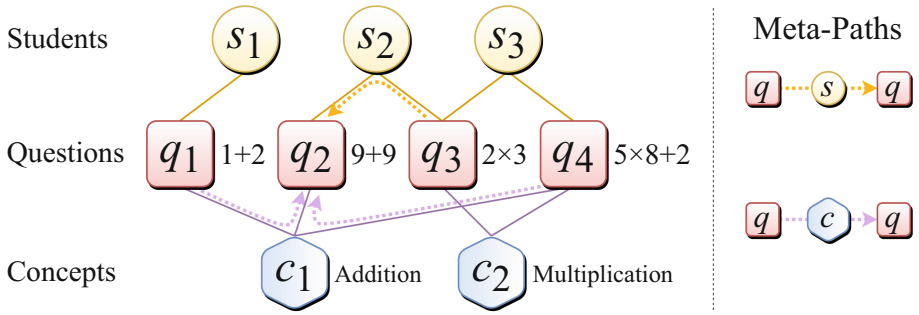

Fig. 1. An example of knowledge tracing data, student-question-concept relation graph, and meta-paths. Questions q_3 and q_4 are done by the same student s_3, and they have similar related concepts and difficulties. Questions q_1 and q_2 are related to the same concept c_1 but have different difficulties. Question q_2's meta-based neighbors include q_2, q_3 (via q-s-q), q_1, and q_4 (via q-c-q).

contain three main types of relations: student-question, question-concept, and question-answer. Existing KT methods utilize the knowledge in these entities and relations with varying degrees. However, they do not model all types of relations in a comprehensive and uniform way.

For question and concept entities, many KT models [19,33] only use concepts as the input to learn a student's concept mastery, and they ignore the specific information and the student's state of each question, causing the loss of latent information between them [1,30] (see q_1, q_2 in Fig. 1). For student entities, few KT models consider using student information as the input, and students are only indices of sequence. Using student information is meaningful, and the questions done by the same student can implicitly reflect that their learning stages and difficulties are similar. Some previous work partially utilizes relational information by introducing a question-concept graph [15,27,30] or student-specific parameters [20,31], but they do not fully use all types of relations.

We believe that the KT task can be better achieved by comprehensively and uniformly modeling all types of entities while retaining specific feature information. Since these entities with different types have unique effects and structural information, complete modeling of all entities and their relations can best extract the rich information contained therein.

Inspired by previous work [9,11], we represent all types of entities and relations in KT as vertices and edges in a heterogeneous graph. A heterogeneous graph [21,23] contains multiple types of vertices and edges, which can naturally express data with complex structures in the real world. As shown in Fig. 1 and 2, the heterogeneous graph demonstrates these entities and their relations. In addition, heterogeneous graph neural networks (HGNNs) [8,28] can obtain semantic relational information in a heterogeneous graph.

In this paper, we propose a deep KT model, namely **Heterogeneous Graph Representation for Knowledge Tracing** (HGRKT), to model various types

of entities and relations in the KT task in a unified way. We first construct a *knowledge tracing heterogeneous graph* (KTHG), which describes the student-question-concept-answer relations. Then, we use a meta-path-based multi-layer HGNN to obtain heterogeneous relation-aware representations in KTHG. Specifically, we use Heterogeneous Graph Attention Network (HAN) to implement HGNN, which is based on hierarchical attention and performs well among HGNNs. After that, we feed the representations of questions into the KT model, and the whole model can be optimized through end-to-end backpropagation.

In summary, our main contributions are as follows:

1. We construct a KTHG to comprehensively and uniformly model the interactive and structural relations among students, questions, concepts, and answers, where each vertex has an embedding and represents an entity with a type. Our proposed method is simple and concise by exploiting the high representational capacity of the heterogeneous graph.
2. We apply HGNN to get representations for KT. HGNN can effectively extract the rich knowledge contained in relations into the generated question embeddings, and the embeddings are used as the input of the KT model.
3. Experiments on multiple benchmark datasets show our model's effectiveness. Our method improves AUC by 1% compared to the best baseline on average.

2 Related Work

2.1 Knowledge Tracing

Existing KT models can be roughly divided into non-deep models and deep models. In recent years, deep learning has been widely used in KT task research for its powerful ability to extract and represent features and discover complex structures [14]. DKT [19] first introduces deep learning to KT and uses a Recurrent Neural Network (RNN) [29] to model a student's knowledge state. DKVMN [33] uses Memory Augmented Neural Network to automatically discover basic concepts and trace the state of each concept. Since then, various deep KT methods have been proposed based on the two methods.

Many researchers introduce graph and question features to model knowledge structure in different ways, such as utilizing concept graphs [3,17,25], considering question features and combining them with concept features in prediction [16,27], and using question-concept relation graphs and graph neural networks (GNNs) [13] to learn embeddings for questions [15,30].

2.2 Heterogeneous Graph

A heterogeneous graph [21,23] is a graph with multiple types of vertices and edges, which can naturally describe complex relations and rich semantics in many data mining tasks [9,11]. Recently, researchers have proposed many Heterogeneous graph neural networks (HGNNs) models [8,28,32] to extract representations of vertices and obtain knowledge in heterogeneous graphs.

Our method mainly uses the Heterogeneous Graph Attention Network (HAN) [28]. HAN is an HGNN based on vertex-level and semantic-level attentions. Vertex-level attention uses a semantic-specific GAT [26] layer to learn the importance between vertices and their meta-path-based neighbors, while semantic-level attention measures the importance of meta-paths with an attention vector and fuses embeddings of all meta-paths to obtain the final embedding.

3 Preliminaries

3.1 Knowledge Tracing

In the KT task, a group of students $\mathcal{S}$ sequentially answer a series of questions from a set $\mathcal{Q}$. At the time step t, a student $s \in \mathcal{S}$ answers a question $q_t \in \mathcal{Q}$, and the correctness of the answer is $a_t \in \{0, 1\}$. The observed learning interaction sequence before the time step T is

$$\boldsymbol{X} = \{q_t, a_t\}_{t=1}^{T} = \{(q_1, a_1), \ldots, (q_t, a_t), \ldots, (q_T, a_T)\}. \tag{1}$$

In the time step $T + 1$, given the historical sequence $\boldsymbol{X}$ and a new question q_{T+1}, the goal of KT is to predict the probability of the student correctly answering the new question:

$$\hat{a}_{T+1} = P(a_{T+1} = 1 | \boldsymbol{X}, q_{T+1}). \tag{2}$$

Let ℓ be the loss function (e.g., binary cross-entropy). The predictive loss of the answer is $\ell(a_{t+1}, \hat{a}_{t+1})$, and the loss for a student is $\sum_t \ell(a_{t+1}, \hat{a}_{t+1})$.

3.2 Heterogeneous Graph

A *heterogeneous graph* [23] $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a vertex set $\mathcal{V}$ and an edge set $\mathcal{E}$, and it is associated with a vertex type map and an edge type map so that each vertex and edge has its own type.

A *meta-path* [24] Φ is a path $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \cdots \xrightarrow{R_l} A_{l+1}$ describing a composite relation of all intermediate edge types $R_1, \ldots, R_l$ between vertex types A_1 and A_{l+1}. Given a meta-path Φ, the *meta-path-based neighbors* $\mathcal{N}^{\Phi}$ is a map from vertex i to a set of vertices that connect with vertex i via meta-path Φ. Meta-path is a fundamental structure of a heterogeneous graph, which can express certain semantics and reveal diverse structure information.

As shown in Fig. 1, the meta-path question-concept-question represents questions with the same concept (e.g., q_1 and q_4), while the meta-path question-student-question represents questions done by the same student (e.g., q_2 and q_3). Question q_2's neighbors based on the meta-path question-student-question include q_2 (itself) and q_3, indicating that s_2 answers both. Its neighbors based on meta-path question-concept-question include q_2, q_1, and q_4, indicating that they share the same concept c_1.

3.3 Message Passing

To utilize the graph structural knowledge, we apply the message passing to get the new embeddings of vertices in graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$. We can use a GNN layer to apply the message passing from source vertices $\{v_i|(v_i, v_j) \in \mathcal{E}\}$ to target vertices $\{v_j|(v_i, v_j) \in \mathcal{E}\}$ via all edges. We construct an embedding matrix $\mathbf{M} = (\mathbf{h}_1^{\mathrm{T}}; \ldots; \mathbf{h}_N^{\mathrm{T}}) \in \mathbb{R}^{N \times d}$, consisting of all vertices' embedding vectors, where $\mathbf{h}_i = \mathbf{M}_{i,*} \in \mathbb{R}^d$ is vertex v_i's embedding vector. The GNN transforms the input embedding matrix into a new embedding matrix $\mathbf{M}' = \mathrm{GNN}(\mathbf{M}|\mathcal{G})$, where the new embedding of vertex i is $\mathbf{h}'_i = \mathbf{M}'_{i,*}$.

For a heterogeneous graph $\mathcal{G}'$, we should construct multiple embedding matrices for each type of vertices. For vertices with type A_k, its corresponding embedding matrix is M_k. The message passing is applied from source vertices to target vertices with various types. Let the source vertices have n_s different types $\{A_1^{\mathcal{S}}, \ldots, A_{n_s}^{\mathcal{S}}\}$ and target vertices have n_t different types $\{A_1^{\mathcal{T}}, \ldots, A_{n_t}^{\mathcal{T}}\}$, then the HGNN transforms the embedding matrices of sources into embedding matrices of targets, i.e., $\{M_1^{\mathcal{T}}, \ldots, M_{n_t}^{\mathcal{T}}\} = \mathrm{HGNN}(M_1^{\mathcal{S}}, \ldots, M_{n_s}^{\mathcal{S}}|\mathcal{G}')$.

4 Method

4.1 KTHG

The data of knowledge tracing includes students, questions, concepts, answers, and their relations. We model them as vertices and edges with different types in a *knowledge tracing heterogeneous graph* (KTHG). Let $\mathcal{S}$, $\mathcal{Q}$, and $\mathcal{C}$ be the set of students, questions, and concepts separately. Let $\mathcal{A} = \{a|a = 0 \text{ or } a = 1\}$ be the set of answers, where the member 1 and 0 indicates whether the answer is correct or not. We define the KTHG $\mathcal{G}_{\mathrm{KT}}(\mathcal{V}_{\mathrm{KT}}, \mathcal{E}_{\mathrm{KT}})$ as the combination of three simpler heterogeneous graphs $\mathcal{G}_{sq}, \mathcal{G}_{qc}, \mathcal{G}_{qa}$, i.e., $\mathcal{G}_{\mathrm{KT}} = \mathcal{G}_{sq} \cup \mathcal{G}_{qc} \cup \mathcal{G}_{qa}$.

Student-Question Graph. The student-question graph $\mathcal{G}_{sq}(\mathcal{S} \cup \mathcal{Q}, E_{sq})$ is a bipartite graph, where E_{sq} contains the interactions between students and questions in the training data, and an undirected edge (s_i, q_j) means that student $s_i \in \mathcal{S}$ answers question $q_j \in \mathcal{Q}$, as is shown in the upper part of Fig. 2(b).

Question-Concept Graph. The question-concept graph $\mathcal{G}_{qc}(\mathcal{Q} \cup \mathcal{C}, E_{qc})$ is a bipartite graph, where E_{qc} contains the relations between questions and concepts. An undirected edge (q_i, c_j) means that question $q_i \in \mathcal{Q}$ is related to concept $c_j \in \mathcal{C}$, as is shown in the lower part of Fig. 2(b).

Question-Answer Graph. The question-answer graph $\mathcal{G}_{qa}(\mathcal{Q} \cup \mathcal{A} \cup \mathcal{Q}_A, E_{qq_a} \cup E_{aq_a})$ is a combination of two complete bipartite graphs $\mathcal{G}_{qq_a}(\mathcal{Q} \cup \mathcal{A}, E_{qq_a})$ and $\mathcal{G}_{aq_a}(\mathcal{A} \cup \mathcal{Q}_A, E_{aq_a})$, where $\mathcal{Q}_A$ is the set of questions combined with answers, and $\mathcal{Q}_A = \{q_{i0}, q_{i1}\}_{i=1}^{|\mathcal{Q}|}$ for $\mathcal{Q} = \{q_i\}_{i=1}^{|\mathcal{Q}|}$, and $|\mathcal{Q}_A| = 2|\mathcal{Q}|$. Concretely, to combine the question $q_i \in \mathcal{Q}$ with answer a, we use two new vertices $q_{i1}, q_{i0} \in \mathcal{Q}_A$ to

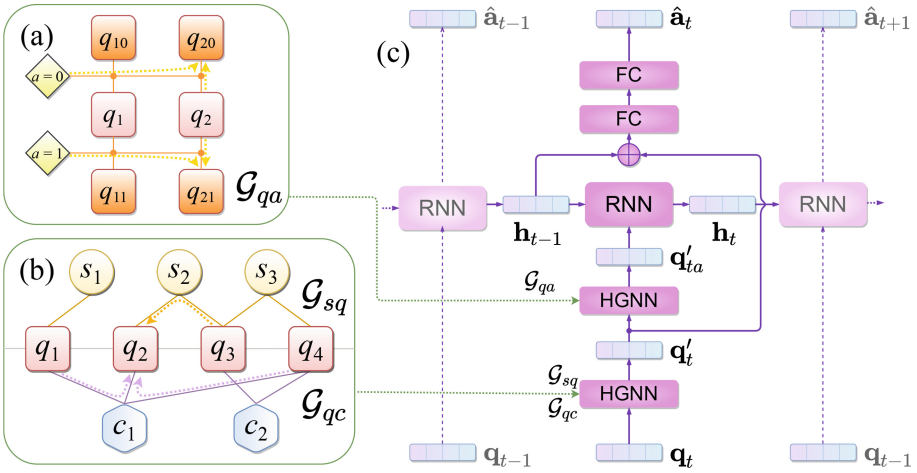

Fig. 2. The architecture of our method. (a) The question-answer graph $\mathcal{G}_{qa}$; (b) The student-question graph $\mathcal{G}_{sq}$ and the question-concept graph $\mathcal{G}_{qc}$; (c) The overview of our method. It shows the updating of a student's knowledge state in the time step t. Two parts of KTHG in (a) and (b) are used in two HGNN layers in (c) separately. Dotted lines show the message passing via meta-paths.

represent the question with the correct and incorrect answer respectively, called question-with-answer (qa for short) vertices.

As shown in Fig. 2(a), question q_1 and two answers, "$a = 0$" and "$a = 1$", are linked into q_{10} and q_{11} by two meta-paths, question-qa and answer-qa. We use these two meta-paths to fuse a question embedding and an answer embedding into a question-with-answer embedding.

4.2 Representation Learning on KTHG

We stack the HGNN layers to make up a multi-layer HGNN, and apply message passing on KTHG with a two-layer HGNN to generate representations of vertices with graph structure information about other vertices connected with them. Although we may get the new embeddings for any type of vertices, we only use generated embeddings about questions as the input feature of the KT model because students' performance is mainly related to them.

For all types of vertice sets $\mathcal{S}, \mathcal{Q}, \mathcal{C}, \mathcal{A}, \mathcal{Q}_A$, we construct embedding matrices $\mathbf{M}_S, \mathbf{M}_Q, \mathbf{M}_C, \mathbf{M}_A, \mathbf{M}_{QA}$. At first, We use an HGNN layer to transform the original question embeddings, student embeddings, and concept embeddings into aggregated question embeddings:

$$\mathbf{M}'_Q = \mathrm{HGNN}_{sqc}(\mathbf{M}_S, \mathbf{M}_Q, \mathbf{M}_C | \mathcal{G}_{sq} \cup \mathcal{G}_{qc}), \tag{3}$$

where $\mathbf{M}'_Q = \left(\mathbf{q}'^{\mathrm{T}}_1; \ldots; \mathbf{q}'^{\mathrm{T}}_N\right)$, $\mathbf{q}'_i = \left(\mathbf{M}'_Q\right)_{i,*}$ and $N = |\mathcal{Q}|$. Here, we apply message passing on the student-question-concept graph and use meta-paths ended with

the question vertex, and the aggregated question embeddings carry information about questions, students, and concepts.

When modeling the student's learning process, we need to combine the information about questions and answers. We use another HGNN layer to fuse an aggregated question embeddings $\mathbf{M}'_Q$ and an answer embeddings $\mathbf{M}_A$ into an aggregated question-with-answer embeddings $\mathbf{M}'_{QA}$:

$$\mathbf{M}'_{QA} = \mathrm{HGNN}_{qa}(\mathbf{M}'_Q, \mathbf{M}_A | \mathcal{G}_{qa}), \tag{4}$$

where $\mathbf{M}'_{QA} = \left(\mathbf{q}'^{\mathrm{T}}_{10}; \mathbf{q}'^{\mathrm{T}}_{11}; \ldots; \mathbf{q}'^{\mathrm{T}}_{N0}; \mathbf{q}'^{\mathrm{T}}_{N1}\right)$ and $\mathbf{q}_{ia} = \left(\mathbf{M}'_{QA}\right)_{2i+a-1,*}$. Here, we apply message passing on the question-answer graph and use the meta-paths ended with the question-with-answer vertex, so the aggregated question-with-answer embeddings carry the information of its corresponding aggregated question embeddings and answer embeddings.

4.3 KTHG Representations for KT Model

We trace a student's knowledge state by his/her learning interaction sequence $\{q_t, a_t\}_{t=1}^{T}$. Note that here q_t denotes the question ID in the time step t (instead of the question with ID t). The architecture of our KT model is shown in Fig. 2(c). After representation learning on KTHG, we get question embedding matrices $\mathbf{M}'_Q, \mathbf{M}_{QA}$ with graph structure information for the KT model. For question q_t, we can get its aggregated question embedding $\mathbf{q}'_t = (\mathbf{M}'_Q)_{q_t,*}$. When tracing the student's knowledge state, we can get the question-with-answer embedding $\mathbf{q}'_{ta} = (\mathbf{M}'_{QA})_{2q_t+a_t-1,*}$ for the current answer a_t.

Then, similar to DKT [19], we use an RNN to trace the student's knowledge state. For each time step t, we feed the embedding $\mathbf{q}'_{ta}$ into an RNN cell to update the RNN's hidden state vector $\mathbf{h}_t$:

$$\mathbf{h}_t = \mathrm{RNN}(\mathbf{q}'_{ta}, \mathbf{h}_{t-1}), t = 1, 2, \ldots, T. \tag{5}$$

In the prediction, we use the student's current knowledge state and question embedding to predict the answer. Specifically, we first concatenate RNN's current state $\mathbf{h}_{t-1}$ with the aggregated question embedding $\mathbf{q}'_t$. After that, we feed the concatenated vector into a fully connected layer to obtain the summary vector $\mathbf{h}'_t$, and then we feed the vector into a Sigmoid activation layer to calculate the probability of correctly answering, denoted as $\hat{a}_t$:

$$\mathbf{h}'_t = \tanh(\mathbf{W}_{fc}(\mathbf{h}_{t-1} \oplus \mathbf{q}'_t) + \mathbf{b}_{fc}), \tag{6}$$

$$\hat{a}_t = \sigma(\mathbf{W}_{\mathfrak{a}}\mathbf{h}'_t + \mathbf{b}_{\mathfrak{a}}), \tag{7}$$

where $\oplus$ denotes concatenation operator and σ is the Sigmoid function.

5 Experiment

5.1 Datasets

We evaluate our method on four open public datasets: ASSIST09, ASSIST12, EdNet, and Junyi, which are sampled from educational platforms [2,6,10].

Table 1. Dataset statistics.

Statistics	ASSIST09	ASSIST12	EdNet	Junyi
#Records	185,110	1,839,429	326,267	622,781
#Students	2,968	22,422	4,700	7,000
#Questions	15,003	45,543	11,060	1,978
#Concepts	121	99	189	39
#Concepts per Question	1–4	1	1–7	1
Avg. #Questions Per Concept	150.8	460.0	128.7	50.7
AVg. #Concepts Per Question	1.2	1.0	2.2	1.0
Avg. #Attempts Per Question	12.3	40.4	29.5	314.9
Avg. #Attempts Per Concept	1,914.2	18,580.1	4,026.9	15,968.7
Correct Rate (%)	63.8	69.6	59.7	67.3

We filter out records without concepts and students with less than 10 interactions and randomly sample students to make computing resources affordable. The statistics of the four datasets are shown in Fig. 1. We use the indices of questions, their related concepts, and students' answers from records. The maximum length of students' interaction sequence is set to 200. In the datasets, We split 4/5 of students for training and validation, and another 1/5 for testing, where no student appears in both training and testing sets.

5.2 Implementation Details

To evaluate the effectiveness of our method, we compare our method with three groups of 11 KT baseline methods. Single-state methods represent a student's knowledge state with a vector, which is often the hidden state of RNN; Multi-state methods represent the student's knowledge concept state with multiple vectors; State-free methods do not maintain a vector to represent the student's knowledge state. Note that our method is single-state.

To fairly compare the results of all methods, we uniformly set some shared parameters. The dimension of question embeddings and concept embeddings is 64. The batch size is 16. We initialize the parameters randomly, and the optimizer is Adam [12] with a learning rate of 0.001. We use a single-layer GRU [4] to implement RNN [4], and the dimension of its hidden state is 64. We run each method five times and show their average results of AUC.

We choose HAN [28] as the HGNN layer implementation, called **HAN-HGRKT**. In HAN, the dimension of the semantic-level attention vector is 128. The input dimension of the vertex embedding is set to 64. Besides, we only update part of useful vertices in the current batch to reduce memory usage. We pick five meaningful meta-paths for two HAN layers: (1) q-s-q, q-c-q, c-q for $\mathcal{G}_{sq}, \mathcal{G}_{qc}$; (2) q-$qa$, a-$qa$ for $\mathcal{G}_{qa}$, where "q", "s", "c", "a", and "qa" denote question, student, concept, answer, and question-with-answer respectively. The code of our implementation is available at https://github.com/chenjisen/HGR-KT.

Table 2. AUC results (%) on four datasets.

Group	Method	ASSIST09	ASSIST12	EdNet	Junyi
Single-state	DKT [19]	67.1	68.8	70.2	88.5
	EERNNA [22]	74.1	74.1	71.5	86.9
	DHKT [27]	74.2	74.1	71.6	89.0
	GIKT [30]	75.5	73.9	72.5	88.9
Multi-state	DKVMN [33]	65.6	68.6	68.9	88.5
	GKT [17]	75.1	68.5	69.1	85.8
	SKT [25]	71.3	66.4	66.4	86.1
State-free	SAKT [18]	68.9	69.5	70.5	79.5
	CKT [20]	70.3	70.0	64.9	87.7
	SKVMN [1]	64.7	68.2	68.9	88.2
	SAINT [5]	68.5	68.5	69.9	85.3
(Ours)	HAN-HGRKT *	**76.8**	**74.4**	**73.4**	**89.5**

Table 3. Attention weights of meta-paths in two HAN layers.

Graphs	Meta-path	ASSIST09	ASSIST12	EdNet	Junyi
$\mathcal{G}_{sq}, \mathcal{G}_{qc}$	q-s-q	0.39	0.64	0.53	0.52
	q-c-q	0.35	0.36	0.29	0.36
	c-q	0.26	0.00	0.18	0.12
$\mathcal{G}_{qa}$	q-qa	0.71	0.81	0.69	0.66
	a-qa	0.29	0.19	0.31	0.34

5.3 Results

We measure AUC to evaluate and compare the performance of methods. All decimals are rounded to the nearest tenth. Table 2 shows AUC results in experiments, and a higher AUC indicates that the method predicts student performances better. We observe that our method, HAN-HGRKT, outperforms all 11 baselines and achieves state-of-the-art performance in all four datasets, demonstrating that our method leverages all types of relations better with the HAN. Our method outperforms the best baseline (GIKT) by about 1% in AUC on average. In addition, there is no significant difference in performance between state-free, single-state, and multi-state methods.

However, our method has the least improvement on ASSIST12 and Junyi, which only outperforms the best baseline for no more than 1% in AUC. We assume that in these two datasets, a question is only related to a concept, so they are simpler than other datasets, and there is no place for the heterogeneous graph to show its powerful representation capacity.

Besides, we get the attention weight values of all meta-paths in HAN. Table 3 shows the results, where a mid-line separates the results from two HAN layers. We find that the HAN can automatically adjust the weight of each meta-paths on different datasets.

Table 4. AUC results (%) of HAN-HGRKT with different graphs and meta-paths in two HAN layers.

Graphs	Meta-paths	ASSIST09	ASSIST12	EdNet	Junyi
$\mathcal{G}_{sq}$	q-s-q	75.1	74.4	73.3	**89.5**
$\mathcal{G}_{qc}$	c-q	76.0	68.4	69.3	85.6
	q-c-q	76.3	**74.5**	71.7	89.1
	q-c-q, c-q	76.6	**74.5**	71.8	89.0
$\mathcal{G}_{sq}, \mathcal{G}_{qc}$	q-s-q, q-c-q	76.4	74.4	**73.4**	**89.5**
	q-s-q, q-c-q, c-q*	**76.8**	74.4	**73.4**	**89.5**
$\mathcal{G}_{qa}$	q-qa, a-qa *	**76.8**	74.4	**73.4**	**89.5**
	q-qa, aq-qa	71.4	71.7	70.9	86.8
(AE)	(AE)	76.7	74.4	73.2	89.4
(ZV)	(ZV)	76.5	74.3	73.2	89.1

5.4 Ablation Study

We design the ablation study to further analyze the effect of each sub-graph and meta-path in our method.

Student-Question Graph and Question-Concept Graph. We change the usage of $\mathcal{G}_{sq}$, $\mathcal{G}_{qc}$, and their related meta-paths in the first HAN layer, while not changing the second HAN layer. The upper part of Table 4 shows the results, where our original method performs the best, and removing any graphs or meta-paths decreases the performance. We notice that the meta-path c-q has the least effect, but it has the most significant improvement and the largest attention weight (see Table 3) on ASSIST09, indicating that the c-q relation contains irreplaceable semantic information on ASSIST09 compared to the other datasets, and our method can automatically catch it.

Question-Answer Graph. We compare our second HAN layer with other methods of combining questions with answers, while not changing the first HAN layer. The variants of the methods of combining answers with questions are listed as follows: (1) Using a new vertex type "aq", which means that we use two answer vertices for each question respectively, and the number of "aq" vertices is $2|\mathcal{Q}|$; (2) AE: concatenating new question embedding with answer embedding instead of using $\mathcal{G}_{qa}$, i.e., $\mathbf{q}'_{ta} = \mathbf{q}'_t \oplus \mathbf{a}_t$; (3) ZV: concatenating new question embedding with zero vectors in two orders instead of using $\mathcal{G}_{qa}$, i.e., $\mathbf{q}'_{ta} = \mathbf{q}'_t \oplus \mathbf{0}$ when $a_t = 1$ and $\mathbf{0} \oplus \mathbf{q}'_t$ when $a_t = 0$. The lower part of Table 4 shows the results, where our original method performs the best. We notice that if we construct two answer embedding for each question vertex individually, the performance decreases a lot, and the reason may be that it increases the complexity of the model and causes difficulty for training.

6 Conclusion

In this paper, we propose an end-to-end method, Heterogeneous Graph Representation for Knowledge Tracing (HGRKT), to learn question representations for the KT task. We introduce a KTHG to model heterogeneous relations among students, questions, concepts, and answers. Then, we use a multi-layer HGNN to obtain the question representations with rich relational knowledge. After that, we use a KT model to get the students' performances. We evaluate our method on four KT datasets and compare it with 11 knowledge tracing baseline methods. As our model gets better question representations, it performs state-of-the-art performance and outperforms the best baseline by 1% on average. The extensive ablation study shows the effectiveness of the whole KTHG with the meta-paths. In the future, we will apply different implementations of heterogeneous graph models to explore a better extraction method of heterogeneous relations.

Acknowledgments. This work is supported by Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102) and National Natural Science Foundation of China (62177033).

References

1. Abdelrahman, G., Wang, Q.: Knowledge tracing with sequential key-value memory networks. In: Proceedings of SIGIR (2019)
2. Chang, H.S., Hsu, H.J., Chen, K.T.: Modeling exercise relationships in e-learning: a unified approach. In: Proceedings of EDM (2015)
3. Chen, P., Lu, Y., Zheng, V.W., Pian, Y.: Prerequisite-driven deep knowledge tracing. In: Proceedings of ICDM (2018)
4. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of EMNLP (2014)
5. Choi, Y., et al.: Towards an appropriate query, key, and value computation for knowledge tracing. In: Proceedings of L@S (2020)
6. Choi, Y., et al.: Ednet: a large-scale hierarchical dataset in education. In: Proceedings of AIED (2020)
7. Corbett, A.T., Anderson, J.R.: Knowledge tracing: modeling the acquisition of procedural knowledge. In: Proceedings of UMUAI (1994)
8. Dong, Y., Chawla, N.V., Swami, A.: metapath2vec: Scalable representation learning for heterogeneous networks. In: Proceedings of KDD (2017)
9. Fan, S., et al.: Metapath-guided heterogeneous graph neural network for intent recommendation. In: Proceedings of KDD (2019)
10. Feng, M., Heffernan, N., Koedinger, K.: Addressing the assessment challenge with an online system that tutors as it assesses. In: Proceedings of UMUAI (2009)
11. Gao, W., et al.: RCD: Relation map driven cognitive diagnosis for intelligent education systems. In: Proceedings of SIGIR (2021)
12. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: Proceedings of ICLR (2015)
13. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: Proceedings of ICLR (2017)
14. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature (2015)

15. Liu, Y., Yang, Y., Chen, X., Shen, J., Zhang, H., Yu, Y.: Improving knowledge tracing via pre-training question embeddings. In: Proceedings of IJCAI (2020)
16. Minn, S., Desmarais, M.C., Zhu, F., Xiao, J., Wang, J.: Dynamic student classification on memory networks for knowledge tracing. In: Proceedings of PAKDD (2019)
17. Nakagawa, H., Iwasawa, Y., Matsuo, Y.: Graph-based knowledge tracing: modeling student proficiency using graph neural network. In: WI (2019)
18. Pandey, S., Karypis, G.: A self-attentive model for knowledge tracing. In: Proceedings of EDM (2019)
19. Piech, C., et al.: Deep knowledge tracing. In: Proceedings of NeurIPS (2015)
20. Shen, S., et al.: Convolutional knowledge tracing: modeling individualization in student learning process. In: Proceedings of SIGIR (2020)
21. Shi, C., Li, Y., Zhang, J., Sun, Y., Yu, P.S.: A survey of heterogeneous information network analysis. In: Proceedings of TKDE (2017)
22. Su, Y., et al.: Exercise-enhanced sequential modeling for student performance prediction. In: Proceedings of AAAI (2018)
23. Sun, Y., Han, J.: Mining heterogeneous information networks: a structural analysis approach. ACM SIGKDD Explor. Newsl. (2013)
24. Sun, Y., Han, J., Yan, X., Yu, P.S., Wu, T.: PathSim: Meta path-based top-k similarity search in heterogeneous information networks. Proc. VLDB Endow. (2011)
25. Tong, S., et al.: Structure-based knowledge tracing: an influence propagation view. In: Proceedings of ICDM (2020)
26. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: Proceedings of ICLR (2018)
27. Wang, T., Ma, F., Gao, J.: Deep hierarchical knowledge tracing. In: Proceedings of EDM (2019)
28. Wang, X., et al.: Heterogeneous graph attention network. In: Proceedings of WWW (2019)
29. Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural Comput. (1989)
30. Yang, Y., et al.: GIKT: a graph-based interaction model for knowledge tracing. In: Proceedings of ECML-PKDD (2020)
31. Yudelson, M.V., Koedinger, K.R., Gordon, G.J.: Individualized Bayesian knowledge tracing models. In: Proceedings of AIED (2013)
32. Zhang, C., Song, D., Huang, C., Swami, A., Chawla, N.V.: Heterogeneous graph neural network. In: Proceedings of KDD (2019)
33. Zhang, J., Shi, X., King, I., Yeung, D.Y.: Dynamic key-value memory networks for knowledge tracing. In: Proceedings of WWW (2017)