Contextualized Knowledge Graph Embedding for Explainable Talent Training Course Recommendation

YANG YANG, Nanjing University of Science and Technology and Hong Kong Polytechnic University
CHUBING ZHANG, Nanjing University of Science and Technology
XIN SONG and ZHENG DONG, Baidu Talent Intelligence Center, Baidu Inc
HENGSHU ZHU, Career Science Lab, BOSS Zhipin
WENJIE LI, Hong Kong Polytechnic University

Learning and development, or L&D, plays an important role in talent management, which aims to improve the knowledge and capabilities of employees through a variety of performance-oriented training activities. Recently, with the rapid development of enterprise management information systems, many research efforts and industrial practices have been devoted to building personalized employee training course recommender systems. Nevertheless, a widespread challenge is how to provide explainable recommendations with the consideration of different learning motivations from talents. To this end, we propose CKGE, a contextualized knowledge graph (KG) embedding approach for developing an explainable training course recommender system. A novel perspective of CKGE is to integrate both the contextualized neighbor semantics and high-order connections as motivation-aware information for learning effective representations of talents and courses. Specifically, in CKGE, for each entity pair (i.e., the talent-course pair), we first construct a meta-graph, including the neighbors of each entity and the meta-paths between entities as motivation-aware information. Then, we develop a novel KG-based Transformer, which can serialize entities and paths in the meta-graph as a sequential input, with the specially designed relational attention and structural encoding mechanisms to better model the global dependence of KG structured data. Meanwhile, the local path mask prediction can effectively reveal the importance of different paths. As a result, CKGE not only can make precise predictions but also can discriminate the saliencies of meta-paths in characterizing corresponding preferences. Extensive experiments on real-world and public datasets clearly validate the effectiveness and interpretability of CKGE compared with state-of-the-art baselines.

CCS Concepts: • **Information System → Data Ming;**

Additional Key Words and Phrases: Course recommendation, knowledge graph, Transformer

This work was supported by the National Key RD Program of China (2022YFF0712100), the NSFC (61906092, 62006118, 62276131), the Natural Science Foundation of Jiangsu Province of China (grant BK20200460), the Jiangsu Shuangchuang (Mass Innovation and Entrepreneurship) Talent Program, the Young Elite Scientists Sponsorship Program by CAST, and the Fundamental Research Funds for the Central Universities (NJ2022028 and 30922010317).
Authors' addresses: Y. Yang, Nanjing University of Science and Technology and Hong Kong Polytechnic University, China; email: yyang@njust.edu.cn; C. Zhang, Nanjing University of Science and Technology, China; email: chubing_zhang@njust.edu.cn; X. Song and Z. Dong, Baidu Talent Intelligence Center, Baidu Inc, China; emails: songxin06@baidu.com, zhdong0@outlook.com; H. Zhu (corresponding author), Career Science Lab, BOSS Zhipin, China; email: zhuhengshu@gmail.com; W. Li (corresponding author), Hong Kong Polytechnic University, China; email: cswjli@comp.polyu.edu.hk.

1046-8188/2023/09-ART33 $15.00
https://doi.org/10.1145/3597022

ACM Reference format:
Yang Yang, Chubing Zhang, Xin Song, Zheng Dong, Hengshu Zhu, and Wenjie Li. 2023. Contextualized Knowledge Graph Embedding for Explainable Talent Training Course Recommendation. *ACM Trans. Inf. Syst.* 42, 2, Article 33 (September 2023), 27 pages.
https://doi.org/10.1145/3597022

1 INTRODUCTION

As a long-standing topic in human resource management, learning and development, or L&D, aims at improving the knowledge and capabilities of employees through a variety of performance-oriented training activities, which plays an important role for companies to build their competitive edge in the knowledge-based economy [33]. Recently, with the rapid development of enterprise management information systems, there has been a trend in companies to build internal online course platforms for facilitating the active learning of employees. In these platforms, employees can choose different courses to learn for various purposes. As a result, extensive learning records and talent profiles have been accumulated, which provide an unprecedented opportunity for researchers to study the L&D problems through a data-driven paradigm [22, 29, 42, 53, 55, 56].

Indeed, a popular research topic along this line is to build personalized employee training course recommender systems. Although considerable research efforts and industrial practices have been devoted in this direction, a widespread challenge is how to provide explainable recommendations considering different learning motivations of talents [9, 24, 41, 42]. Different from traditional recommendation scenarios (e.g., movies or product recommendations), the learning motivation of talents may be influenced by a variety of underlying factors, such as current competencies, learning histories, and the preference of colleagues. For example, a *Java engineer* who has close collaboration with algorithm teams might also like to learn machine learning courses. Meanwhile, an *algorithm engineer* who has learned *Deep Learning Foundation* may continue to learn *Advanced Deep Learning Technique* due to the dependency and correlation between courses. Therefore, it is vitally important to include the extra motivation-aware information for reasonably modeling the representations of talents and courses, so as to address individual needs and recommend more meaningful courses. However, these complicated motivations are difficult to interpret in existing recommendation models with only talent-course interaction data.

Inspired by the recent success of **Knowledge Graph (KG)**-based recommender systems [14], we find that these potential factors can be formalized as the contextual information of the talent-course pair in the talent-course KG. Therefore, in this article, we propose to use talent-course KG as auxiliary information for enhancing the performance and interpretability of personalized training course recommendations. Specifically, the talent-course KG contains various types of information, where the entities include the talents, skills, or courses, and relations can be represented as various patterns. Therefore, the neighbors of talents/courses and the meta-paths between talent-course pairs in the graph can efficiently provide explainable insights for corresponding recommendation results. State-of-the-art KG-based recommendation models [5, 27, 37, 51, 61, 62] mainly extend the traditional techniques [1] with entity similarity derived from KG representations, which cannot effectively integrate the neighbor information and sophisticated connection patterns into consideration. Therefore, some works [37, 48] focused on the joint modeling, which leverages the idea of embedding propagation to refine the representations of entities and inherits the interpretability from meta-paths.

In this work, we propose CKGE, a contextualized KG embedding approach for developing an explainable training course recommender system, which can learn effective representations of heterogeneous KG entities by integrating both of the neighbor semantics and high-order connections.

Specifically, in CKGE, we first construct a meta-graph, including the neighbors of each entity and the meta-paths between entities, for each talent-course pair as motivation-aware information. Then, we develop a novel KG-based Transformer network, called *KGformer*, for learning the representations. KGformer includes the following. The first is the *relational attention mechanism*. We specially design a relational attention mechanism in KGformer to capture the structural relation between nodes. This mechanism can help the model accurately capture the dependency in a meta-graph by assigning a mask matrix based on spatial relations as masked self-attention. The second is *structural encoding*. Inspired by Ying et al. [58], we also exploit the edge-based structural encoding to capture the relation embedding in the attention module, which can better model the neighbors and connection patterns for the entity pairs in representation learning. The third is *local path mask prediction*. In addition to traditional interactive prediction, we employ mask prediction on the path level to determinate the importance of different meta-paths. As a result, CKGE can globally consider the contributions of neighbors and paths to learn entity representations, so it not only makes precise **Click-Through Rate (CTR)** prediction but also characterizes corresponding preferences. The contributions of this work are summarized as follows:

- We proposeCKGE for talent training course recommendation, which designs a novel KG-based Transformer for learning the embedding of entities better and describing the learning motivation.
- We develop KGformer considering the structural information by relational self-attention and edge encoding.
- We empirically show the superiority of CKGE on real-world course recommendation data and a public dataset.

In the following sections, we first introduce related work in Section 2 and then give the details of our proposed method in Sections 3 and 4. In Section 5, we conduct comprehensive experiments to validate the effectiveness of our proposed method. We conclude the article in Section 6.

2 RELATED WORK

In this article, we incorporate the KG into the recommendation system, and thereby the proposed method is related to knowledge representation learning and knowledge-aware recommendation.

2.1 Knowledge Representation Learning

KG is a heterogeneous graph, in which the nodes are entities and edges denote relations between entities. In addition, KG can be decomposed of subject-property-object triple facts. Each triplet can be formally defined as (h, r, t), where head/tail (i.e., h/t) is entity and r is relation. Knowledge representation learning aims at learning the representations of entities and relations while preserving the inherent structure [21]. The straightforward solutions are distance-based approaches, which calculate the distance between relational projections of entities. For example, TransE [3] directly constrained the added embedding of head and relation to be close to the embedding of tail; RotatE [36] defined each relation as a rotation from the source entity to the target entity in the complex vector space; and Wang et al. [46] took a sequence as input to obtain contextualized representations, which are hence naturally capturing contextual meanings of entities and relations therein. To date, KGs have been created and applied in multiple scenarios, including natural language understanding [8], question answering [17], and recommendation systems [62].

2.2 KG-Based Recommendation

Deep learning based algorithms have achieved significant success in the recommendation literature. Actually, our method belongs to the user-item recommendation group [63], which aims to

predict a user's preferred items directly. Considering the effectiveness for addressing the sparsity and cold-start problems, the usage of a KG for better recommendation has attracted increasing attention.

At first, many works developed embedding-based regularization to improve recommendation. For example, Zhang et al. [62] proposed collaborative KG embedding, which learns better item latent representations by capturing entity semantics via TransR [25]; Wang et al. [43] proposed formulating the user recommendation task as the sentiment link prediction task between entities in the graph; Cao et al. [5] developed a translation-based user preference model to integrate with KG learning, which jointly learned the task of recommendation and KG completion; and Ye et al. [57] learned a low-dimensional representation for various entities by integrating the multi-modal information via a neural factorization machine. These methods leverage the KG to enrich the representations of users and items, whereas they have difficulty in capturing high-order relations between entities. Therefore, Hu et al. [19] and Ma et al. [27] introduced learning the representations of meta-paths to depict the interaction context of user-item pairs; Ma et al. [27] proposed modeling of connection patterns of associated items (co-buy, co-view, etc.) in an external item KG into rule features; and Wang et al. [49] leveraged rich semantics in meta-graphs for user state representation, then trained the candidate generation model to promote an efficient search in the action space. Furthermore, several attempts have been made to integrate both entity/relation representation learning and high-order connection pattern learning for fully exploiting the information in a KG. For example, Wang et al. [45] proposed KGCN, in which the weight of each neighbor is defined by the type of user for aggregation; Qu et al. [30] proposed a neighborhood-based interaction model, which considered the interactions between item-side neighbors and user-side neighbors; and Liu et al. [26] exploited both local and non-local graph context information of an entity in the KG for recommendation. However, these approaches always aim to learn more meaningful representations, which are difficult to elaborately discover the recommendation motivation.

2.3 Explainable Recommendation

Explainable recommendation can simultaneously provide suitable recommendations and explanations for users to understand why they are recommended those items [38, 64]. Recently, knowledge-aware explainable recommendation [20, 44, 62] has been leveraged for explainable recommendation since the knowledge base contains rich external structural information of users and items. Early attempts included KG representation based approaches, which usually combine knowledge-aware representation with item representations to generate a better representation for item. For example, Zhang et al. [62] exploited the knowledge base by jointly learning the latent representations in collaborative filtering as well as items' semantic representations from the knowledge base, and Wang et al. [44] proposed a multi-channel and word-entity-aligned knowledge-aware convolutional neural network that fuses semantic-level and knowledge-level representations. However, the explanations of why the items are recommended are weak because the introduced KG representations are implicit. Therefore, meta-path-based methods have been proposed [13, 38, 60], where the paths between a user and recommended items in the KG are considered as explanations. These methods extend the general entity similarity derived from paths, which can extract structural features to capture relevant semantics for recommendation. For example, Sun et al. [37] proposed a recurrent KG embedding approach to mine the paths relation between users and items automatically; Wang et al. [48] modeled each intent as an attentive combination of KG relations and recursively integrated the relation sequences of long-range connectivity; and Tan et al. [38] proposed a novel path language modeling recommendation, which learned a language model over KG paths to unify recommendation and explanation through path sequence decoding. However,

the meta-path based methods rely heavily on the quality of selected meta-paths, which are difficult to elaborately measure the dependence between neighbors, paths, and the talent-course pair, which in turn may affect representation learning and motivation discovery. Therefore, we turn to explore the meta-graph between a user and recommended items in the KG, which can learn more discriminative user/item representations and extract more meaningful structures to capture relevant semantics for recommendation.

3 DATA DESCRIPTION

In this section, we mainly introduce the real-world dataset we used, which is in-firm data of talents from a high-tech company in China (note that all sensitive information in the dataset has been removed or anonymized). Actually, the KG is constructed from talent-course interaction records (i.e., the courses that users have learned). Specifically, the constructed KG contains 3 types of entities: talent, skill, and course (including 16,438 talents, 6,504 professional skills, and 7,337 courses). The skill profile indicates the professional job skills that talent has already mastered. Meanwhile, the KG owns 13 types of relationships: 3 types of relationships between talents (including leader, peer, and cooperation) based on the organizational structure, project cooperation and other factors; 2 types of relationships between courses (including after and before) considering that a domain content is usually divided into several courses similar to student education [66]; 5 types of relationships between talents and skills (including skill_l1 to skill_l5, which stands for basic mastery to mastery); 1 type of relationship between talents and courses (i.e., learned); and 2 types of relationships between skills and courses (including dominate and contain) considering the degree of correlation. To this end, the constructed KG contains 201,005 talent-relation-talent facts, 202 course-relation-course facts, 165,372 talent-relation-course facts, 1,658,905 talent-relation-skill facts, and 14,192 course-relation-skill facts. It is notable that the collected talent-relation-course facts have noisy interactions considering that talents may only click the online course but do not spend time learning. Thereby, we consider a fact as a valid record only when the course is studied for more than half of the time following [42]. In addition, the talent-relation-course facts (learning interaction records) have a common problem among most recommendation systems—that is, the data sparse problem that most courses are studied by a small number of talents (only 0.19% of the talent-course pairs own valid relationships).

4 PROPOSED METHOD

The key challenges of talent training course recommendation lie in the accuracy and explainability. Therefore, we aim to fully exploit the auxiliary information encoded in talent-course KG for learning more discriminative user/item representations, as well as the motivation for recommendation. Different from traditional meta-path-based approaches, we turn to construct a more meaningful meta-graph between a user and the recommended item, which includes both the paths and neighbors to provide more auxiliary information. Furthermore, to encode more discriminative user/item representations, we propose KGformer, which leverages Transformer to better mine the contextual semantics with the newly designed relational self-attention and encoding. Consequently, we can learn motivation-aware representations of both talents and courses, which can be used to generate better recommendations, and we can also mine the semantic meanings of relation patterns (i.e., the meta-paths) in KG, which can be further utilized to capture the explanation of talent-course matching.

4.1 Preliminary

Notations. We denote the talent set as $\mathcal{U} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\}$, the skill set as $\mathcal{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_o\}$, and the course set as $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_n\}$. The historical talent-course interaction records can be

denoted as $\mathbf{I} \in \mathbb{R}^{m \times n}$, where $I_{i,j} = 1$ indicates that $\mathbf{u}_i$ has learned $\mathbf{c}_j$ and 0 otherwise. Without any loss of generality, we utilize the relation "learned" to express $I_{i,j} = 1$. However, we can use the entity as a generic term to refer all relevant objects (i.e., talent, skill, and course) and utilize relation as the connection between entities.

Knowledge Graph. Let $\mathcal{E} = \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{ke}\}$ denote the set of entities, including the talent, skill, and course nodes ($ke = m + o + n$), and let $\mathcal{R} = \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{kr}\}$ represent the set of relations. Therefore, KG can be formulated as subject-property-object triple facts—that is, $T = \{(\mathbf{e}_h, \mathbf{r}_{ht}, \mathbf{e}_t)\}$, $\mathbf{e}_h$ is the head entity, $\mathbf{r}_{ht}$ is the relation, and $\mathbf{e}_t$ is the tail entity.

Transformer. Given the long sequence $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l\} \in \mathbb{R}^{l \times d}$ with length l, where d is the hidden dimension and $\mathbf{x}_i \in \mathbb{R}^d$ denotes the hidden representation at position i. Traditional Transformer [39] aims to encode the relationships among independent token by adopting the multi-head attention mechanism. In detail, the identical block contains two sub-layers: (1) the first sub-layer utilizes multi-head attention to learn the correlated representations, and (2) the second sub-layer adopts a position-wise feed-forward network. In the multi-head attention layer, the input representations can be used to compute three matrices: Q, K, and V corresponding to queries, keys, and values. The dot-product similarity between queries and keys determines attention distributions:

$$\begin{aligned} &Q = XW_Q, \quad K = XW_K, \quad V = XW_V, \\ &A = \frac{QK^\top}{\sqrt{d_M}} \quad Att(X) = \sigma(A)V, \end{aligned} \tag{1}$$

where $Q \in \mathbb{R}^{l \times d_M}$, $K \in \mathbb{R}^{l \times d_M}$, $V \in \mathbb{R}^{l \times d_M}$, and $W_Q \in \mathbb{R}^{d \times d_M}, W_K \in \mathbb{R}^{d \times d_M}, W_V \in \mathbb{R}^{d \times d_M}$ are learnable matrices. The activation function σ can be used as softmax here. Multi-head attention is composed of M parallel heads, and $d_M = d/M$. Results of each head are concatenated and passed through a linear transformation to construct the output—that is, $MultiAtt(X) = [Att(X)_1, \ldots, Att(X)_M]W_M$, where $W_M \in \mathbb{R}^{d \times d}$ is the learnable parameter. The feed-forward network is a fully connected network, which can be formulated as follows:

$$FFN(MultiAtt(X)) = \max(0, MultiAtt(X)W_1 + b_1)W_2 + b_2, \tag{2}$$

where W_1 and W_2 are matrices for linear transformation, and b_1 and b_2 are the bias terms. Meanwhile, each sub-layer is followed by dropout [34], shortcut connection [16], and layer normalization [2]. We can also add the special token $[CLS]$ for learning the global representations. Finally, we can acquire the global embedding from the $[CLS]$ token and individual embedding from other tokens. In the following section, we specifically introduce the KG-based Transformer for encoding the contextualized knowledge information into the entity embedding.

4.2 Meta-Graph Construction

The traditional recommendation technique (i.e., collaborative filtering) always suffers from sparsity of user-item interactions and the cold-start problem, and it is difficult to explore interpretability. To alleviate these problems, we incorporate auxiliary KG into the embedding learning. Typically, compared with KG-free methods, KG-based talent-course recommendation benefits the results in two ways. First, the various types of linked neighbors are helpful for extending talent and course representations reasonably. For example, as shown in Figure 1, Jerry's superior and his colleagues are all deep learning algorithm engineers, and he has also learned deep learning courses; thus, we can better represent Jerry's portrait through the fusion of neighbor embedding. Second, KG connects abundant talent-course relations, which can explain learning motivation better than simple talent-course interaction. For example, as shown in Figure 1, compared to the simple interaction $Jerry \xrightarrow{learned} Python$, the meta-path $Jerry \xrightarrow{leader} Jack \xrightarrow{skill\_15} Data\ Analysis \xrightarrow{dominate} Python$

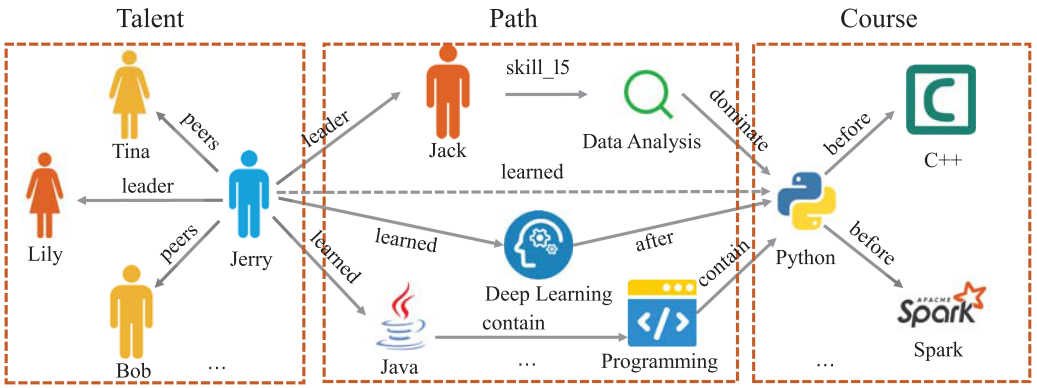

Fig. 1. Example of a talent-course meta-graph. "(Jerry, Python)" is the given talent-course pair, and we can extend the pair into a meta-graph (including neighbors and meta-paths) with the auxiliary KGs.

is more meaningful since the path can further mine the learning motivation that Jerry's leader (i.e., Jack) is a senior technician. He has just taken a course on Python, so considering the needs of project and other factors, there is a high probability that colleagues in the same group will learn the same course. Based on this idea, to fully exploit the motivation-aware entity neighbors and relations in talent-course KG, we first mine the meta-graph for a given entity pair—that is, the neighbors of head/tail entity and paths with different semantics between entities—which can be seamlessly fused into the encoder for effective recommendation.

Neighbor Sampling. Consider a candidate talent-course pair (i.e., $(\mathbf{u}, \mathbf{c})$); we utilize $\mathcal{N}(\mathbf{e_u})$ and $\mathcal{N}(\mathbf{e_c})$ (because $\mathbf{u}$ and $\mathbf{c}$ are also entities in KG) to denote the set of entities directly connected to $\mathbf{e_u}/\mathbf{e_c}$, $\{\mathbf{r}_{ij} | i = \mathbf{u}, j \in \mathcal{N}(\mathbf{e_u})\}$ denotes the relation between $\mathbf{e}_i$ and $\mathbf{e}_j$ of the talent's facts, and the same for course's facts. In the real-world KG, the size of $\mathcal{N}(\mathbf{u})$ and $\mathcal{N}(\mathbf{c})$ may vary significantly over different entities. To keep the computational pattern fixed and more efficient, we sample a fixed amount of neighbors (i.e., $|\mathcal{N}(\mathbf{u})| = |\mathcal{N}(\mathbf{c})|$) for each entity instead of using its full neighbors as done by Wang et al. [45].

Meta-Path Sampling. Except for the direct linkage between the talent-course pair, meta-paths connecting the entity pair carry more relational dependencies for representing the relational roles of entities. Thereby we also sample paths between the talent-course pairs to mine contextualized information. To increase model efficiency, we use random walks to sample paths as done by Perozzi et al. [28]. In detail, we adopt the biased random walks to obtain the probability distribution of next entities by the following equation:

$$P(\mathbf{e}_{i+1} | \mathbf{e}_i) = \begin{cases} \frac{\omega_{i,i+1}}{\sum_{\mathbf{e}_j \in \mathcal{N}(\mathbf{e}_i)} \omega_{i,j}}, & \exists (\mathbf{e}_i, \mathbf{r}_{i,i+1}, \mathbf{e}_{i+1}) \in T \\ 0, & otherwise \end{cases} \tag{3}$$

where $\omega_{i,j} = exp(-(num(r_{i,j}) + num(\mathbf{e}_j))/\tau)$, $num(a)$ denotes the amount of entity/relation a, τ is the temperature scale, and $\mathbf{e}_j$ is selected from all neighbors of $\mathbf{e}_i$. The biased random walks choose the next entities considering the entity/relation imbalanced problem, and thereby we can take the rare facts into account. In fact, there exist a large number of meta-paths connecting entity pairs, which contain different entity and relation types in different orders and with various lengths.

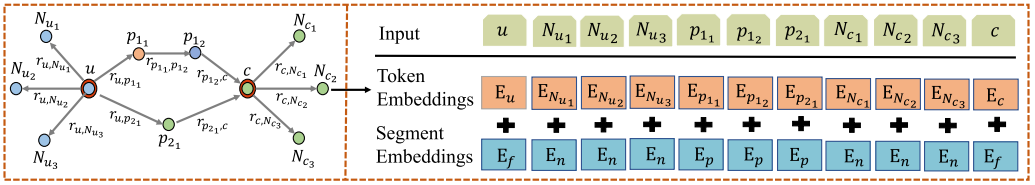

Fig. 2. Example of meta-graph serialization. In the meta-graph shown on the left, blue dots represent talents, yellow dots denote skills, and green dots are courses. We serialize entities in the meta-graph as tokens, using token embeddings and segment embeddings.

However, not all paths contribute to KG's embedding, as longer paths may bring in remote neighbors and lose semantic meanings [35]. Therefore, we enumerate paths with a length constraint—that is, only paths with the length less than a threshold are employed.

Guided by these stages, CKGE can mine qualified neighbors and paths with abundant semantics that connect entity pairs in a meta-graph pattern instead of manually designed and extracted features from the KG. The meta-graph will be further processed by the developed KG-based Transformer to automatically learn entity and path semantic representations.

4.3 KG-Based Transformer (KGformer)

In this section, we present our KGformer for the motivation-aware meta-graph encoding. We first give the sequence construction including the neighbors and paths of the entity pair. Then, we elaborate on several key designs in the KGformer—that is, the relational self-attention, relation encoding, and path mask prediction, which leverage the structural information of the KG into the Transformer model to learn the representations.

Serialization of a Meta-Graph. In the traditional Transformer, the input data X in Equation (1) is naturally sequential (e.g., time series data, long sentence data). However, the entities in the meta-graph are not arranged in sequence. Therefore, the difficulty lies in how to effectively sort entity pairs, relations, neighbors, and paths in the meta-graph. In KGformer, we use truncated serialization, which temporarily ignores the relations and adds segment embeddings as an additional signal to the input. To be specific, as shown in Figure 2, the meta-graph contains three neighbors for the head/tail entity and two paths connecting the entity pair. Thereby, we regard the head entity and the tail entity as the start token and the end token of the sequence, respectively, then arrange the neighbors and path entities in order. We also add the segment embeddings to refer to the category of tokens—that is, fact (i.e., E_f), neighbor (i.e., E_n), and path (i.e., E_p). In the end, we acquire the serialized tokens to represent the entities in the meta-graph.

Relational Self-Attention. The advantage of Transformer is the global receptive field by the self-attention mechanism, which needs to specify positions (i.e., absolute positional encoding [39]) or encodes the positional dependency (i.e., relative positional encoding [31, 58]). Nevertheless, entities in the meta-graph have no concept of sequence, and it makes no sense that each token attends to the information at any position considering the relational linkages. Therefore, to encode the structural information between entities, we propose relational self-attention, which times the spatial distance with the self-attention results for structural attention. Specifically, as shown in Figure 3, we develop a spatial distance function $\phi(\mathbf{e}_i, \mathbf{e}_j)$ to measure the spatial relation between $\mathbf{e}_i$ and $\mathbf{e}_j$. In this work, we define the $\phi(\mathbf{e}_i, \mathbf{e}_j) = \mu_{d(\mathbf{e}_i, \mathbf{e}_j)}$, where $d(\mathbf{e}_i, \mathbf{e}_j)$ denotes the length of the shortest path between $(\mathbf{e}_i, \mathbf{e}_j)$ and $\mu_{d(\mathbf{e}_i, \mathbf{e}_j)}$ is a learnable parameter for each $d(\mathbf{e}_i, \mathbf{e}_j)$. Thereby, the A_{ij} in Equation (1) can be reformulated as

$$\hat{A}_{ij} = \phi(\mathbf{e}_i, \mathbf{e}_j) \frac{(\mathbf{e}_i W_Q)(\mathbf{e}_j W_K)^\top}{\sqrt{d_M}}, \tag{4}$$

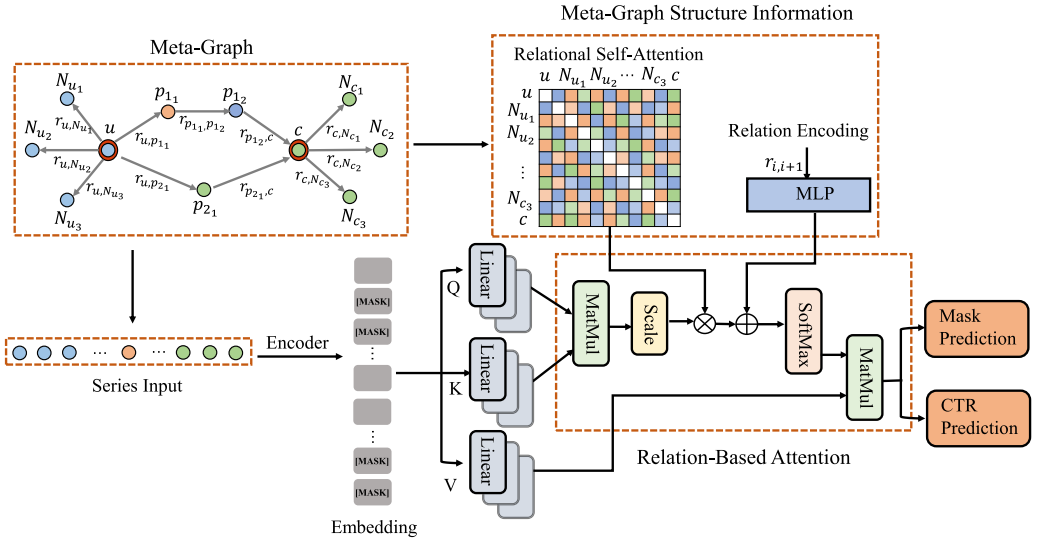

Fig. 3. Illustration of the KG-based Transformer. The schematic diagram is from left to right. We first obtain the input sequence and structure information from the constructed meta-graph, then the sequence embeddings are input to KGformer. Meanwhile, we design the relation-based scale dot-product attention by taking the structural attention into account when calculating the self-attention. Finally, we consider both the path mask prediction and traditional CTR prediction for training.

where $\phi(\mathbf{e}_i, \mathbf{e}_j)$ is shared across all layers. With $\hat{A}_{ij}$, the relational attention will pay more attention to the nearer entities and pay less attention to the remote nodes. As a result, KGformer not only can consider global information with contextualized semantics but also can naturally take structural information into the self-attention.

Relation Encoding. In the talent-course meta-graph, the relations also have semantic information. Therefore, it is necessary to encode the relation features into the entity features. In this work, inspired by Ying et al. [58], we try to adopt relation weighting, which associates the relation embedding together with the entity embedding in the aggregation. In detail, as shown in Figure 3, we develop the relation encoding that incorporates relation embedding via a bias term to the attention module, making the attention mechanism further estimate correlations for each node pair. Concretely, we can modify the attention formulation in Equation (4) as follows:

$$\begin{aligned} \hat{A}_{ij} &= \phi(\mathbf{e}_i, \mathbf{e}_j)\frac{(\mathbf{e}_i W_Q)(\mathbf{e}_j W_K)^\top}{\sqrt{d_M}} + b_{ij}, \\ b_{ij} &= \mathbf{r}_{ij}\omega^\top, \end{aligned} \tag{5}$$

where $\mathbf{r}_{ij} \in \mathbb{R}^d$ is the embedding of relation and $\omega \in \mathbb{R}^d$ is the learnable weight for $\mathbf{r}$ (i.e., the MLP in Figure 3).

4.4 Meta-Path Determination

After encoding with KGformer, we can acquire the embeddings of entities and paths. Assume that there are s paths linking $\mathbf{u}$ and $\mathbf{c}$—that is, $\mathcal{P}(\mathbf{e_u}, \mathbf{e_c}) = \{p_1, p_2, \ldots, p_s\}$ denotes the connected meta-paths, and different paths may play various roles in modeling the motivation. Hence, we introduce the mask prediction in the traditional Transformer [39] to help distinguish the path importance, which is designed to focus on the most important path for predicting the head/tail entity.

Mask prediction utilizes the masked model to learn high-level representations and capture rich relationships between segments. In detail, we only retain the entity $\mathbf{u}$ and one random path p_i from $\mathcal{P}$, and mask the remaining paths and the unselected entity $\mathbf{c}$ with a special token $[MASK]$, then consider predicting the identity of masked entity $\mathbf{c}$ based on context segments. The framework is trained as a conditional masked model:

$$\ell_{mask}(\mathbf{u}, p_i, \mathbf{c}) = -\sum_j y_{\mathbf{c}_j} \log \hat{y}_{\mathbf{c}_j}, \tag{6}$$

where ℓ_{mask} can be any convex loss function, and we utilize the cross entropy for simplicity here. $y_{\mathbf{c}}$ denotes the ground truth of $\mathbf{c}$, and $\hat{y}_{\mathbf{c}} = g(KGformer(\mathbf{u}, p_i))$ denotes the mask prediction, in which the $KGformer(\mathbf{u}, p_i)$ represents the embedding of masked token $\mathbf{c}$ and g is the linear classifier with softmax. Our goal is to predict masked courses, based on the head entity and path via minimizing the negative log-likelihood. It is notable that the importance of different paths is negatively correlated with its own mask loss. Without any loss of generality, we can represent the weight as

$$weight(p_i) = \frac{exp(1/\ell_{mask}(\mathbf{u}, p_i, \mathbf{c}))}{\sum_{p_j \in P} exp(1/\ell_{mask}(\mathbf{u}, p_j, \mathbf{c}))}. \tag{7}$$

The larger the $weight(p_i)$, the more significant this path is.

Finally, based on the talent's representation $\mathbf{e}_\mathbf{u}$ and the course's representation $\mathbf{e}_\mathbf{c}$, the predicted CTR is calculated by

$$\hat{y}_{uc} = \psi(\mathbf{e}_\mathbf{u}^\top \mathbf{e}_\mathbf{c}), \tag{8}$$

where ψ is the sigmoid function. The overall loss function can be represented as

$$L = \sum_{(\mathbf{u},\mathbf{c})} BCELoss(\hat{y}_{uc}, y_{uc}) + \lambda \sum_{p_i \in \mathcal{P}} \ell_{mask}(\mathbf{u}, p_i, \mathbf{c}), \tag{9}$$

where $BCELoss(\cdot)$ is the binary cross entropy [32] between the observed and estimated ratings. λ is the hyper-parameter.

ALGORITHM 1: The pseudo code of CKGE

Input:

- Dataset: Talent-course KG T;
- Parameter: λ;
- maxIter: *Epochs*, learning rate: l_r

```
Initialize KGformer model parameters;
while stop condition is not triggered do
    for mini-batch of talent-course triplet {(e_h, r_ht, e_t)} do
        Meta-graph construction with neighbor sampling, meta-path sampling;
        Serialize constructed meta-graph as input of KGformer;
        Forward propagation with Â_ij according to Equation (5);
        Calculate ℓ_mask according to Equation (6);
        Calculate loss L according to Equation (8);
        Update model parameters using gradient descent;
    end for
end while
```

5 EXPERIMENTS

In this section, we demonstrate the effectiveness of the CKGE framework by verifying the following problems:

- The recommendation performance compared with state-of-the-art baselines
- The performance compared with state-of-the-art baselines under sparse and cold-start scenarios
- Sensitivity analyses of parameters to model performance
- Interpretability of CKGE.

5.1 Baseline Approaches

To verify the effectiveness of CKGE, we compare it with various baseline methods: (1) traditional models (i.e., NCF [18], DeepMF [52], DIEN [65], DCBVN [42], HATCH [54], and DICER [12]), (2) KG-based models (i.e., GEMS [15], TUP [6], KTUP [6], KGAT [47], KGCN [45], KGIN [48], and MetaKG [10]), and (3) heterogeneous graph models (i.e., GraphRec [11], GHCF [7], and PinSage [59]):

- *NCF*: A neural network-based collaborative filtering recommendation method
- *DeepMF*: A collaborative model by implementing matrix factorization with a multi-layer perception network
- *DIEN*: A deep interest evolution network, which designs an interest extractor layer to capture temporal interests from history behavior sequence
- *DCBVN*: A collaborative Bayesian variational network for course recommendation, which jointly models both the employees' competencies and career development preferences
- *HATCH*: A hierarchical adaptive contextual bandit method to conduct the policy learning of contextual bandits with a budget constraint
- *DICER*: A dual-side deep context-aware modulation to capture the friends' information and item attraction
- *GEMS*: A KG recommendation method with genetic meta-structure search, which automatically optimizes the meta-structure designs for recommendation
- *TUP*: A new translation-based recommendation model, which jointly learns the model of recommendation and KG completion
- *KTUP*: A variant of TUP, which jointly learns TUP and TransH [50] to enhance the item and preference modeling by transferring knowledge of entities as well as relations
- *KGAT*: An end-to-end KG attention network, which models the high-order connectivities in KG
- *KGCN*: An end-to-end KG convolutional networks, which captures inter-item relatedness effectively by mining their associated attributes on the KG
- *KGIN*: A KG-based intent network, which models each intent as an attentive combination of relations
- *MetaKG*: A KG-based network, which encompasses a collaborative-aware meta-learner and a knowledge-aware meta-learner, to capture users' preference and entities' knowledge for recommendations
- *GraphRec*: A graph neural network approach, which jointly captures interactions and opinions in the user-item graph
- *GHCF*: A graph convolutional network based model, which captures the high-hop heterogeneous user-item interactions
- *PinSage*: A graph neural network approach, which combines efficient random walks and graph convolutions to generate embeddings of nodes (i.e., items) that incorporate both graph structure and node feature information.

In the baselines, NCF, DeepMF, DIEN, HATCH, and DICER only make use of course records for matrix factorization or sequential modeling. GraphRec, PinSage, and GHCF take the talent-course interaction as a heterogeneous graph, without considering the extra KG. TUP, KTUP, and MetaKG are embedding-based KG models, which use the KG for learning better representations. GEMS is a path-based approach, which depicts the interaction context of talent-course pairs. KGAT, KGCN, and KGIN are propagation-based models, which combine the ideas of path and embedding, and focus on the recommendation interpretability. DCBVN is a specifically designed model for talent-course recommendation by modeling the user learning demands.

5.2 Reproducibility

For the CKGE framework, we apply a grid search in $\{8, 16, 32, 64, 128, 256\}$ to set $d = 256$ with the best performance, and choose the two-layer Transformer network with two heads (i.e., $M = 1$) as the inference network architecture. The neighbor size $|\mathcal{N}(\mathbf{u})| = |\mathcal{N}(\mathbf{c})| = 3$, and the threshold of path length is 3. The hyper-parameter $\lambda = 0.2$ and $\tau = 20$. The optimization method is Adaptive Moment Estimation (Adam), and the learning rate is searched in $\{0.5, 0.1, 0.05, 0.01, 0.005, 0.001\}$ to find the best settings for each task. Finally, we set the learning rate as 0.001. The ratio of dropout is 0.01.

For baseline approaches, we search for the optimal parameters according to the original settings. In detail, the optimal dimension of entity embedding is searched in $\{8, 16, 32, 64, 128, 256\}$. Similarly, a grid search in $\{0.5, 0.1, 0.05, 0.01, 0.005, 0.001\}$ is applied for finding the best learning rate. For GNN-based methods, the number of hidden layers is searched in $\{1, 2, 3\}$, the node dropout rate is searched in $\{0.1, 0.2, \ldots, 0.8\}$, and the hop sampling is selected in $\{1, 2, 3, 4, 5\}$. For KG-based approaches, the number of neighbor, length of path, and other parameters are adjusted as the setting in this work. The coefficient of additional constraint (i.e., L2 regularization) is searched in $\{10e^{-5}, 10e^{-4}, 10e^{-3}, 10e^{-2}, 10e^{-1}\}$ (note that we utilized the normalized output instead of L2 regularization as done by Wang et al. [46]). We run the following experiments with the implementation of an environment on NVIDIA Tesla V100 SXM2 GPUs. The code and dataset will be announced after the work is published.

5.3 Recommendation Performance

To measure performance, we first adopt the widely used evaluation metrics in recommender systems [18, 23, 42, 45]: Recall@N, F1@N, Hit@N, and NDCG@N. Recall@N counts the ratio of successfully predicted items among top-N items to all positive items for each user; F1@N is the harmonic mean of precision at rank N and recall at rank N; Hit@N is 1 if any gold items are recommended within the top-N items, otherwise 0; and NDCG@N represents **Normalized Discounted Cumulative Gain (NDCG)** at N, which places an inverse log reward on all positions that hold a relevant item. The final results reported are the average value of all users. Generally, the larger the values are, the better results acquiring.

First, we conduct experiments under both the normal and sparse settings as done by Wang et al. [42]. In the normal setting, considering the time travel problem (i.e., it is inappropriate to use a talent's future course selection for training and then recommending during the testing process), we select 70% and 10% course records for each user as the training and validation set, respectively, according to the chronological order. The rest of the course records compose the test set. In the sparse setting, the partitioning is similar, but we only chose 30% of courses from each user to form the training set as done by Wang et al. [42].

Table 1 shows the recommendation results of all models in normal and sparse settings. The following can be observed. First, traditional models (i.e., NCF, DeepMF, DCBVN, and HATCH) are behind the best heterogeneous graph method on various criteria, because heterogeneous graph

Table 1. Recommendation Performance of Different Approaches under Normal and Sparse Scenarios

Method	Normal				Sparse			
	Recall@10	F1@10	Hit@10	NDCG@10	Recall@10	F1@10	Hit@10	NDCG@10
NCF	0.0758	0.0380	0.0391	0.0272	0.0258	0.0124	0.0261	0.0130
DeepMF	0.1539	0.0463	0.0782	0.0680	0.1221	0.0312	0.0479	0.0417
DIEN	0.2401	0.0765	0.1555	0.1874	0.1763	0.0486	0.1162	0.1438
DCBVN	0.1570	0.0600	0.1141	0.1522	0.1251	0.0426	0.0815	0.1094
HATCH	0.1953	0.0700	0.1479	0.1665	0.1527	0.0406	0.1021	0.1323
DICER	0.2756	0.0801	0.2632	0.2702	0.2599	0.0741	0.1953	0.2257
GEMS	0.1980	0.0710	0.1510	0.0710	0.1470	0.0379	0.1048	0.0443
TUP	0.2192	0.0783	0.1619	0.1829	0.1769	0.0497	0.1136	0.1487
KTUP	0.2456	0.0801	0.1786	0.1893	0.2003	0.0503	0.1219	0.1546
KGAT	0.2961	0.0853	0.2771	0.2784	0.2782	0.0797	0.2101	0.2364
KGCN	0.2511	0.0831	0.2505	0.2664	0.2311	0.0743	0.2008	0.2123
KGIN	0.3312	0.0912	0.2844	0.3169	0.3021	0.0863	0.2283	0.2526
MetaKG	0.3784	0.1059	0.2795	0.3164	0.3441	0.0968	0.2425	0.3054
GraphRec	0.2269	0.0756	0.1682	0.1832	0.1653	0.0437	0.1021	0.1327
GHCF	0.2459	0.0829	0.2340	0.2263	0.1995	0.0502	0.1728	0.1623
PinSage	0.2577	0.0756	0.2431	0.2467	0.2351	0.0700	0.1893	0.2047
CKGE (w/o RSA)	0.3659	0.1016	0.2757	0.3310	0.3458	0.0943	0.2469	0.3156
CKGE (w/o RE)	0.3760	0.1098	0.2869	0.3448	0.3553	0.0987	0.2553	0.3119
CKGE (Bigger)	0.3651	0.1005	0.2661	0.3240	0.3466	0.0945	0.2445	0.3155
CKGE	**0.3974**	**0.1138**	**0.2991**	**0.3516**	**0.3616**	**0.1025**	**0.2628**	**0.3206**

methods consider more sophisticated neighbor information. Second, the introduction of KG is effective—that is, most KG-based methods (especially propagation-based models) perform better than heterogeneous graph approaches and traditional models on the majority of criteria, because the KG can provide more effective guidance for learning talent/course representations. Third, the performance of the path-based KG model (i.e., GEMS) is not satisfactory, because the generalization of this method is not strong, and the meta-structure found adaptively is not applicable, so this also prompts us to adopt the enumeration strategy in this work. Fourth, propagation-based models achieve better performance on various criteria. This indicates the effectiveness of considering both the neighbors and connection patterns. Fifth, CKGE performs the best on all criteria—for example, CKGE exceeds the best comparison method by around 2% of Recall@10 and 3.4% of NDCG@10 under the normal setting. The results validate the effectiveness of KGformer and meta-graph. In addition, it can be observed that most methods are vulnerable to sparse talents, especially traditional approaches, because traditional approaches learn user preference based on interactions, and thereby they contain very limited information for the sparse problem. We also notice that the decreased ratios of CKGE and most KG-based methods on the sparse setting are less than other methods. This indicates that the constructed meta-graph can provide more information from recommendation KG. Therefore, the recommendation with KG for the sparse problem is useful. The performance of CKGE under the sparse scenario is still optimal (e.g., CKGE exceeds the best comparison method by around 1.7% of Recall@10 and 1.5% of NDCG@10).

5.4 Ablation Study

Table 1 also lists the ablation studies' results of CKGE. CKGE (w/o RSA) represents the variant of CKGE without a relational self-attention module, CKGE (w/o RE) is the variant of CKGE without a relation encoding module, and CKGE (Bigger) denotes the variant of CKGE with deeper Transformer (i.e., an eight-layer Transformer network with four heads as in the work of Ying et al. [58]).

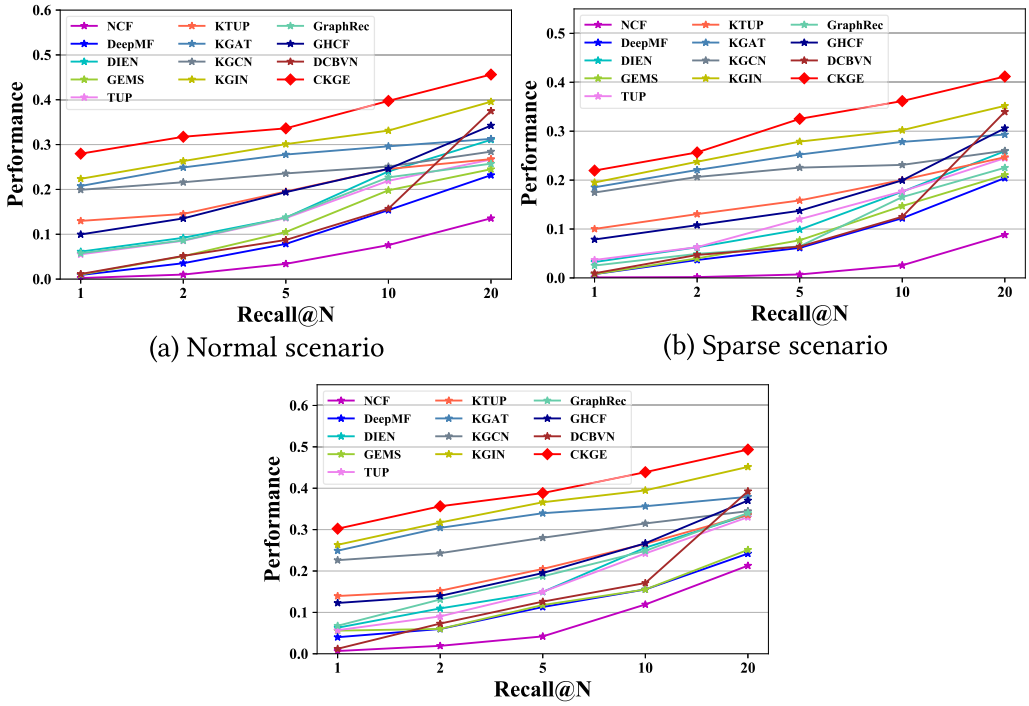

Fig. 4. Recommendation performance of CKGE with various Recall@N.

The following can be observed. First, CKGE (w/o RSA) and CKGE (w/o RE) improve the performance on various criteria, which verifies the contributions of KG- based attention. Second, CKGE (w/o RSA) performs worse than CKGE (w/o RE) on various criteria, which verifies that the relational self-attention module acts in a more important position considering that the dot attention has a greater impact on aggregation. Third, CKGE (Bigger) reduces the performance instead, for the reason that the parameters of the model and the embedding of the entity are updated at the same time, and a more complex network may affect the update of the embedding of the entity. Meanwhile, limited training data is not enough to support the training of larger models.

Considering that Recall@N is a widely used evaluation metric in recommendation systems [42, 45], we add the recommendation performance results by varying the N parameter. We select the typical approaches for comparison here. Figure 4 records the following results. First, CKGE acquires the best performance with different Recall@N under three scenarios, and the performance of CKGE increases faster than other comparison methods. The results further prove the effectiveness of CKGE in recommending performance. Second, CKGE achieves better results under precise measurement (e.g., CKGE acquires 0.2797 of Recall@1 under the normal setting, exceeding other methods at least 5%; CKGE acquires 0.2195 of Recall@1 under the sparse setting, exceeding other methods by at least 2%; and CKGE acquires 0.3022 of Recall@1 under the cold-start setting, exceeding other methods by at least 4%).

5.5 Cold-Start Performance

The cold-start problem is a typical complex problem in recommender systems for which new users have few historical records. According to Sun et al. [37], we set "cold start" as users with fewer than five courses involved in the test data.

Table 2. Recommendation Performance of Different Approaches under the Cold-Start Scenario

Method	Recall@10	F1@10	Hit@10	NDCG@10
NCF	0.1191	0.0253	0.0479	0.0301
DeepMF	0.1554	0.0382	0.0990	0.0732
DIEN	0.2562	0.0471	0.1780	0.2036
DCBVN	0.1710	0.0310	0.1711	0.1809
HATCH	0.2047	0.3870	0.1706	0.1811
DICER	0.3298	0.0699	0.2511	0.2872
GEMS	0.1554	0.0394	0.1698	0.0788
TUP	0.2424	0.0510	0.1883	0.1931
KTUP	0.2656	0.0577	0.1928	0.2190
KGAT	0.3565	0.0733	0.2797	0.3062
KGCN	0.3149	0.0690	0.2746	0.2864
KGIN	0.3948	0.0917	0.2956	0.3320
MetaKG	0.4281	0.0919	0.2963	0.3687
GraphRec	0.2487	0.0557	0.1884	0.2091
GHCF	0.2670	0.0617	0.2423	0.2321
PinSage	0.3059	0.0607	0.2257	0.2536
CKGE (w/o RSA)	0.3878	0.0801	0.2714	0.3371
CKGE (w/o RE)	0.4254	0.1012	0.2977	0.3480
CKGE (Bigger)	0.4183	0.0979	0.2928	0.3451
CKGE	**0.4388**	**0.1087**	**0.3104**	**0.3787**

Table 2 records the performance of all comparison methods. Similar observations with Table 1 can be seen in the cold-start scenario. CKGE significantly outperforms the best existing method—that is, CKGE exceeds the best existing method with 1% and 1% for Recall@10 and NDCG@10, respectively.

5.6 Public Dataset

To validate the generalization of CKGE, we conduct more experiments on a commonly used public dataset: Last.FM.[1] Specifically, Last.FM is a music listening dataset [4], which contains social networking, tagging, and music artist listening information from a set users from the Last.fm online music system. In detail, the dataset has 1,892 users, 17,632 artists, 12,717 bi-directional user friend relations (avg. 13.443 friend relations per user), 92,834 user-listened artist relations (avg. 49.067 artists most listened to by each user and avg. 5.265 users who listened to each artist), and 186,479 tag assignments—that is, user-tag-artist facts (avg. 98.562 tags per user, avg. 14.891 tags per artist, avg. 18.930 distinct tags used by each user, and avg. 8.764 distinct tags used for each artist). In particular, we take the subset of the dataset where the timestamp is from January 2015 to June 2015 following Wang et al. [47], which includes 58,266 entities, nine types of relations, and 464,567 facts. Table 3 and Table 4 record the results, and we find that the experimental results on Last.FM are basically the same as that of the talent-course dataset, which validates the effectiveness of CKGE on the recommendation task.

[1] https://grouplens.org/datasets/hetrec-2011/

Table 3. Recommendation Performance of Last.FM under Normal and Sparse Scenarios

Method	Normal				Sparse			
	Recall@10	F1@10	Hit@10	NDCG@10	Recall@10	F1@10	Hit@10	NDCG@10
NCF	0.0027	0.0047	0.0029	0.0021	0.0008	0.0013	0.0002	0.0012
DeepMF	0.0276	0.0246	0.0156	0.0167	0.0078	0.0047	0.0065	0.0043
DIEN	0.0347	0.0346	0.0248	0.0389	0.0136	0.0156	0.0146	0.0249
DCBVN	0.0293	0.0279	0.0153	0.0270	0.0092	0.0100	0.0042	0.0129
HATCH	0.0310	0.0293	0.0233	0.0376	0.0105	0.0122	0.0127	0.0209
DICER	0.0506	0.0401	0.0291	0.0581	0.0299	0.0270	0.0159	0.0381
GEMS	0.0317	0.0335	0.0246	0.0345	0.0100	0.0115	0.0115	0.0163
TUP	0.0325	0.0337	0.0248	0.0356	0.0116	0.0143	0.0136	0.0234
KTUP	0.0383	0.0351	0.0261	0.0404	0.0192	0.0209	0.0153	0.0291
KGAT	0.0552	0.0460	0.0310	0.0615	0.0345	0.0331	0.0178	0.0433
KGCN	0.0518	0.0421	0.0303	0.0493	0.0316	0.0303	0.0158	0.0310
KGIN	0.0718	0.0679	0.0367	0.0856	0.0529	0.0519	0.0286	0.0677
MetaKG	0.0895	0.1139	0.0579	0.1190	0.0647	0.0917	0.0514	0.0896
GraphRec	0.0308	0.0295	0.0135	0.0311	0.0112	0.0141	0.0054	0.0191
GHCF	0.0403	0.0367	0.0270	0.0474	0.0172	0.0198	0.0120	0.0311
PinSage	0.0471	0.0359	0.0278	0.0511	0.0213	0.0233	0.0147	0.0321
CKGE (w/o RSA)	0.0925	0.1184	0.0628	0.1221	0.0.709	0.0.962	0.0542	0.0934
CKGE (w/o RE)	0.0954	0.1220	0.0645	0.1274	0.0719	0.0970	0.0550	0.0945
CKGE (Bigger)	0.0875	0.1141	0.0626	0.1141	0.0663	0.0893	0.0522	0.0869
CKGE	**0.0966**	**0.1237**	**0.0663**	**0.1285**	**0.0743**	**0.0998**	**0.0565**	**0.0969**

5.7 Comparison with the AUC Metric

Furthermore, we take the AUC to evaluate the model's performance, which is widely adopted in the field of the CTR prediction task. Table 5 records the results on two datasets under various scenarios, and we select the best baselines for comparison. We find that CKGE also performs best under different settings on all datasets, which reveals the effectiveness of CKGE in recommendation.

5.8 Parameter Analysis

To verify the influence of parameters, we conduct more experiments by tuning several important parameters: path length, neighbor number, and loss coefficient. We perform the parameter analyses under normal (No in Figure 6), sparse (S in the figure), and cold-start (C in the figure) scenarios to verify the effectiveness. We select the typical approaches for comparison here. Figure 6 records the performance.

Influence of Path Lengths. Considering that paths with various lengths can capture different semantics, we incorporate paths with different lengths (i.e., $\{2, 3, 4, 5\}$) into the proposed CKGE model to empirically investigate the impact of path length on performance. Figure 6(a), (d), and (g) depict the results; the performance (including Recall@10, F1@10, Hit@10, and NDCG@10) of CKGE first increases, then decreases gradually. CKGE acquires the best performance when length is 3, and the performance difference between length 2 and length 3 is slight. This confirms previous findings that paths with shorter lengths are beneficial for modeling relations as they carry clearer and more interpretable semantic meanings [35]. Meanwhile, we find that an increase in the number of neighbors in the sparse scenario will affect performance more because it is easier to get noisy neighbors in a sparse scenario.

Influence of Neighbor Number. The number of neighbors can also impact the learning of entity embedding. Thereby, we incorporate neighbors with different numbers. Without any loss of generality, we set the same amount of neighbors for talent and course (i.e., $N_u = N_c \in \{1, 2, 3, 4, 5\}$) in the proposed CKGE model to empirically investigate the impact of neighbors on recommendation accuracy. Figure 6(b), (e), and (h) depict the results, in which the performance of CKGE also

Table 4. Recommendation Performance of Last.FM under the Cold-Start Scenario

Method	Recall@10	F1@10	Hit@10	NDCG@10
NCF	0.0104	0.004	0.0103	0.0060
DeepMF	0.0457	0.0125	0.0206	0.0211
DIEN	0.0541	0.0234	0.0263	0.0457
DCBVN	0.0476	0.0155	0.0218	0.0313
HATCH	0.0495	0.0207	0.0319	0.0391
DICER	0.0811	0.0300	0.0784	0.0689
GEMS	0.0495	0.0205	0.0328	0.0416
TUP	0.0502	0.0216	0.0347	0.0429
KTUP	0.0596	0.0259	0.0460	0.0459
KGAT	0.0850	0.0330	0.0830	0.0703
KGCN	0.0802	0.0320	0.0792	0.0578
KGIN	0.1097	0.0421	0.0970	0.0891
MetaKG	0.1166	0.0485	0.1188	0.1352
GraphRec	0.0498	0.0168	0.0313	0.0412
GHCF	0.0643	0.0266	0.0620	0.0503
PinSage	0.0765	0.0259	0.0722	0.0631
CKGE (w/o RSA)	0.1088	0.0437	0.1090	0.1442
CKGE (w/o RE)	0.1121	0.0450	0.1137	0.1455
CKGE (Bigger)	0.1003	0.0419	0.1015	0.1327
CKGE	**0.1282**	**0.0505**	**0.1277**	**0.1462**

Table 5. Recommendation Performance of Two Datasets Using the AUC Criterion

Method	Talent-Course			Last.FM		
	Normal	Sparse	Cold-Start	Normal	Sparse	Cold-Start
Pinsage	0.8756	0.8478	0.8363	0.7837	0.7577	0.7532
DCBVN	0.7195	0.6838	0.6856	0.7333	0.7109	0.7056
KGBIN	0.9195	0.9033	0.8752	0.8571	0.8480	0.8166
MetaKG	0.9277	0.9211	0.8913	0.8706	0.8648	0.8215
GHCF	0.7531	0.7254	0.7136	0.7509	0.7272	0.7154
CKGE	**0.9357**	**0.9289**	**0.8956**	**0.8821**	**0.8701**	**0.8298**

increases first, then decreases on various criteria. The reason may be that more neighbors can even bring noises.

Influence of Loss Coefficient. To explore the influence of each term in Equation (9), we tune the $\lambda \in \{0.1, 0.2, 0.4, 0.6, 0.8\}$ to conduct more experiments. Figure 6(c), (f), and (i) depict the performance of all criteria decreases with the increase of λ, which indicates that the neighbors are more important on the learning of entity embedding.

5.9 Convergence Analysis

Figure 10 further exhibits the convergence versus performance (i.e., Recall@10, Hit@10, NDCG@10, and F1@10) of CKGE. The left vertical axis represents the loss function value, and the right vertical axis is the performance indicator. The horizontal axis represents the number of iterations. We find that CKGE converges fast, and the verification set reaches the optimal result at the 13th epoch on our dataset.

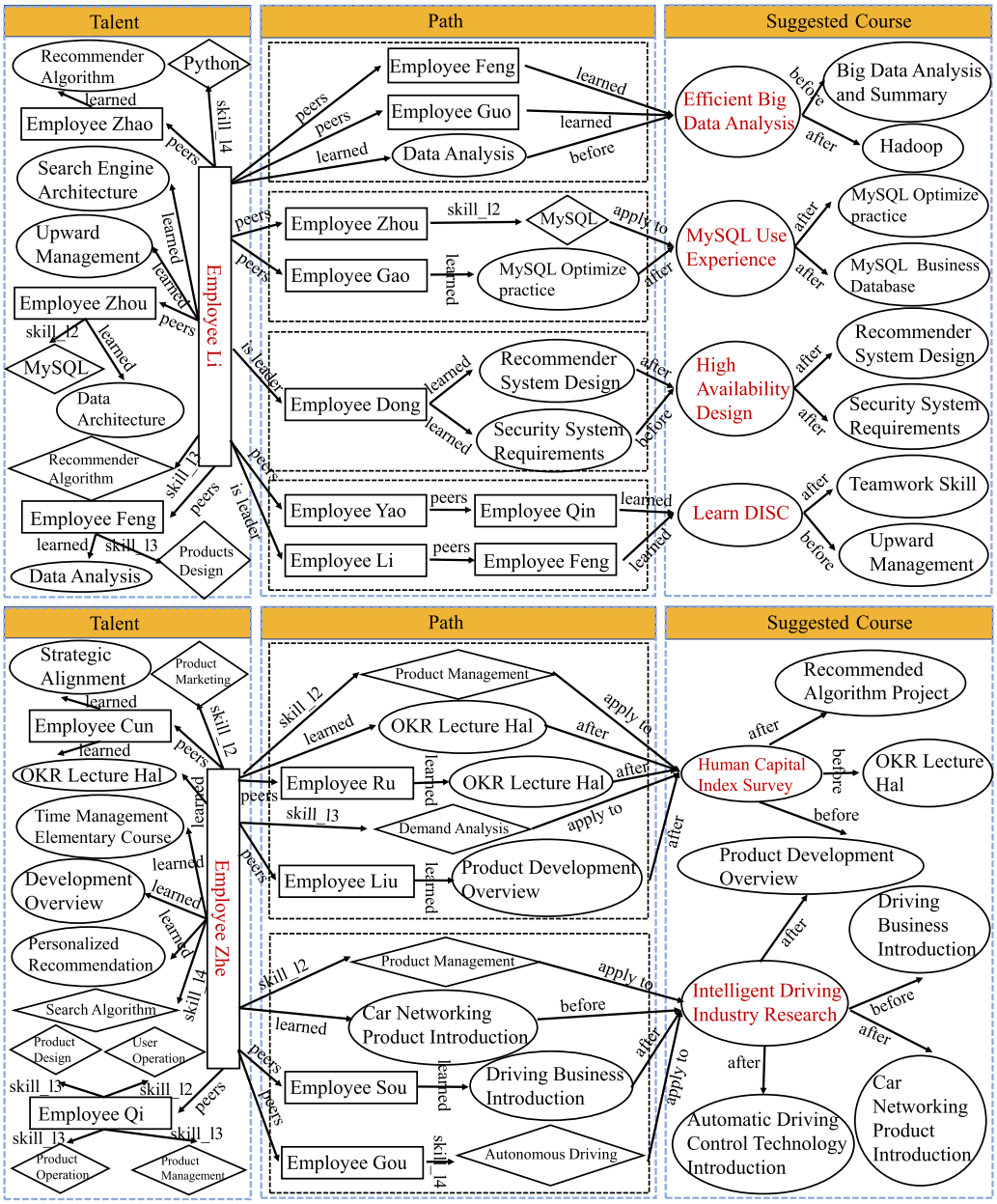

Fig. 5. Prediction examples to illustrate the interpretability of talent-course recommendation with the help of meta-graph information. In detail, we exhibit two examples, and the talents and suggested courses are marked with a red font. The course entity is represented with an ellipse, the skill entity is represented with a diamond, and the talent entity is represented with a cube.

5.10 Statistical Test

To demonstrate the difference between compared approaches and CKGE,we conduct the t-test experiment [40] (i.e., p-value) comparing our method with the comparison approaches. Table 6

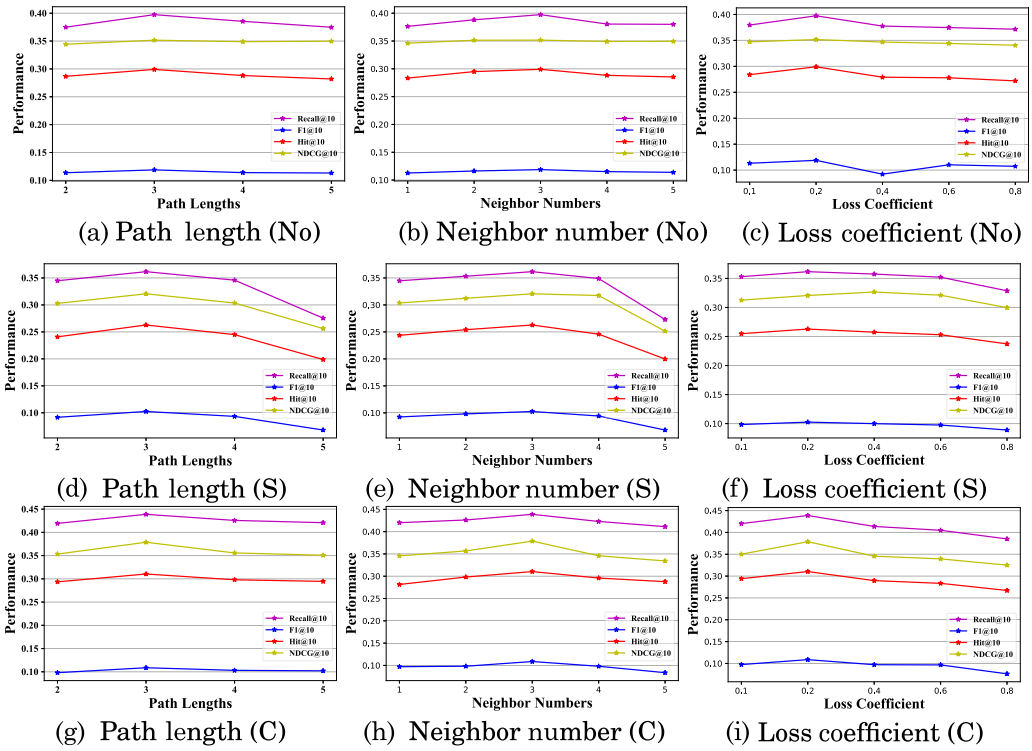

Fig. 6. Parameter analyses of CKGE under normal, sparse, and cold-start scenarios.

Table 6. *t*-Test Performances of Recommendation Performance on the Talent-Course Dataset under Normal and Sparse Scenarios

Method	Normal				Sparse			
	Recall@10	F1@10	Hit@10	NDCG@10	Recall@10	F1@10	Hit@10	NDCG@10
NCF	6.31e-05	3.09e-05	5.30e-05	5.13e-05	7.74e-05	5.85e-05	2.96e-05	5.67e-05
DeepMF	1.77e-05	9.12e-04	7.09e-05	1.70e-05	3.57e-05	2.18e-05	6.19e-05	2.97e-05
DIEN	1.60e-06	1.19e-06	9.96e-05	1.37e-05	9.72e-06	9.36e-06	2.34e-05	5.75e-06
DCBVN	2.96e-05	1.58e-05	1.06e-04	2.89e-05	4.15e-05	2.11e-05	1.26e-04	2.15e-06
GEMS	4.55e-06	1.43e-06	1.37e-05	3.76e-06	8.67e-06	1.94e-05	6.32e-06	3.60e-06
TUP	3.78e-06	2.94e-06	3.85e-05	3.32e-06	8.12e-06	4.67e-05	5.23e-06	5.76e-06
KTUP	1.35e-06	1.54e-06	1.28e-06	1.77e-06	1.66e-06	9.33e-06	5.54e-06	2.91e-06
KGCN	6.24e-05	2.91e-05	4.63e-05	7.89e-05	1.83e-05	1.30e-05	6.60e-05	3.04e-05
KGAT	1.32e-07	3.66e-07	5.24e-05	1.31e-06	2.61e-07	8.84e-07	1.43e-06	2.91e-07
KGIN	3.04e-08	5.85e-08	2.72e-05	4.65e-08	4.11e-07	1.92e-08	7.66e-06	5.94e-07
GraphRec	6.59e-05	2.98e-04	6.12e-04	8.36e-04	2.82e-04	1.81e-04	4.40e-04	4.24e-04
GHCF	1.55e-10	3.59e-08	7.55e-06	5.86e-07	2.19e-07	4.31e-08	7.97e-05	2.35e-07
MetaKG	7.96e-07	2.64e-06	7.03e-05	6.62e-07	6.62e-05	1.46e-05	4.71e-05	1.90e-06
PinSage	2.84e-08	3.40e-07	2.01e-06	2.51e-07	1.84e-07	2.96e-08	3.95e-07	1.95e-07
HATCH	1.32e-05	9.45e-04	4.68e-04	9.67e-05	8.59e-05	5.40e-04	2.59e-03	2.39e-04
DICER	1.83e-05	5.54e-04	5.81e-05	5.07e-05	4.32e-05	4.40e-04	1.87e-04	3.56e-05

Table 7. *t*-Test Performances of Recommendation Performance on the Talent-Course Dataset under the Cold-Start Scenario

Method	Recall@10	F1@10	Hit@10	NDCG@10
NCF	1.14e-05	5.41e-05	4.25e-05	4.38e-05
DeepMF	8.70e-06	2.48e-05	9.86e-04	3.38e-05
DIEN	2.14e-05	2.77e-06	8.77e-04	9.39e-05
DCBVN	3.37e-05	3.13e-05	7.08e-04	3.23e-05
GEMS	2.30e-06	1.33e-06	9.42e-05	6.81e-06
TUP	1.67e-05	8.16e-06	2.67e-05	1.12e-05
KTUP	6.54e-06	7.97e-06	1.20e-05	1.14e-06
KGCN	1.51e-04	6.14e-04	8.82e-04	1.60e-04
KGAT	1.25e-06	7.06e-07	9.23e-05	7.10e-06
KGIN	1.81e-08	3.79e-08	1.75e-05	4.55e-08
GraphRec	2.30e-04	1.02e-04	9.53e-04	2.08e-04
GHCF	3.70e-07	1.66e-07	1.51e-05	3.35e-07
MetaKG	2.06e-05	8.03e-05	2.97e-05	2.15e-05
PinSage	1.24e-08	3.90e-07	2.09e-05	1.48e-08
HATCH	2.86e-05	2.58e-03	5.15e-03	7.55e-04
DICER	3.00e-05	3.62e-04	1.03e-03	2.62e-04

and Table 7 record the results, and we find that our proposed CKGE indeed outperforms other algorithms significantly on normal, sparse, and cold-start scenarios (e.g., the p-values of CKGE are mostly lower than 0.05 compared with other KG-based models).

5.11 Visualization for the Talent-Course Meta-Graph

In Figure 5, we exhibit the interpretability of the talent-course meta-graph. In detail, we first provide the suggested courses for talents (i.e., talent "Employee Li") and suggested courses "Efficient Big Data Analysis, MySQL Use Experience, High Availability Design, and Learn DISC," talent "Employee Zhe," and suggested courses "Intelligent Driving Industry Research and Human Capital Index Survey," then detail the constructed meta-graphs in the KG between the talents and courses according to the proposed meta-graph construction technique. Here we utilize the first talent, Employee Li, as an example. Through the owned skills (e.g., Python, data analysis, recommendation algorithm), learned courses (e.g., machine learning, Hadoop in practice), and neighbors of talent (Employee Zhao who owned skills like Java, Python; Employee Zhou who owned skills like data architecture, big data, etc.), we can clearly conclude that Employee Li is an algorithm engineer who prefers data mining analysis and database management, and has certain experience in recommending system projects. Therefore, we can acquire more accurate talent portraits that are conducive to matching course recommendations. However, the information of "Efficient Big Data Analysis" in the meta-graph shows that this course is generally viewed by data analysis talents and is related to courses such as distributed systems, the information of "MySQL Use Experience" in the meta-graph reveals that the course is widely watched by database background talents, and the information of "High Availability Design" in the meta-graph indicates that the course is suitable for the recommendation system. Similar observations can be obtained for other courses, so as to better match the responding talents. Therefore, talents and courses can be fully characterized through the meta-graphs, resulting in learning motivation-aware representations for making accurate predictions.

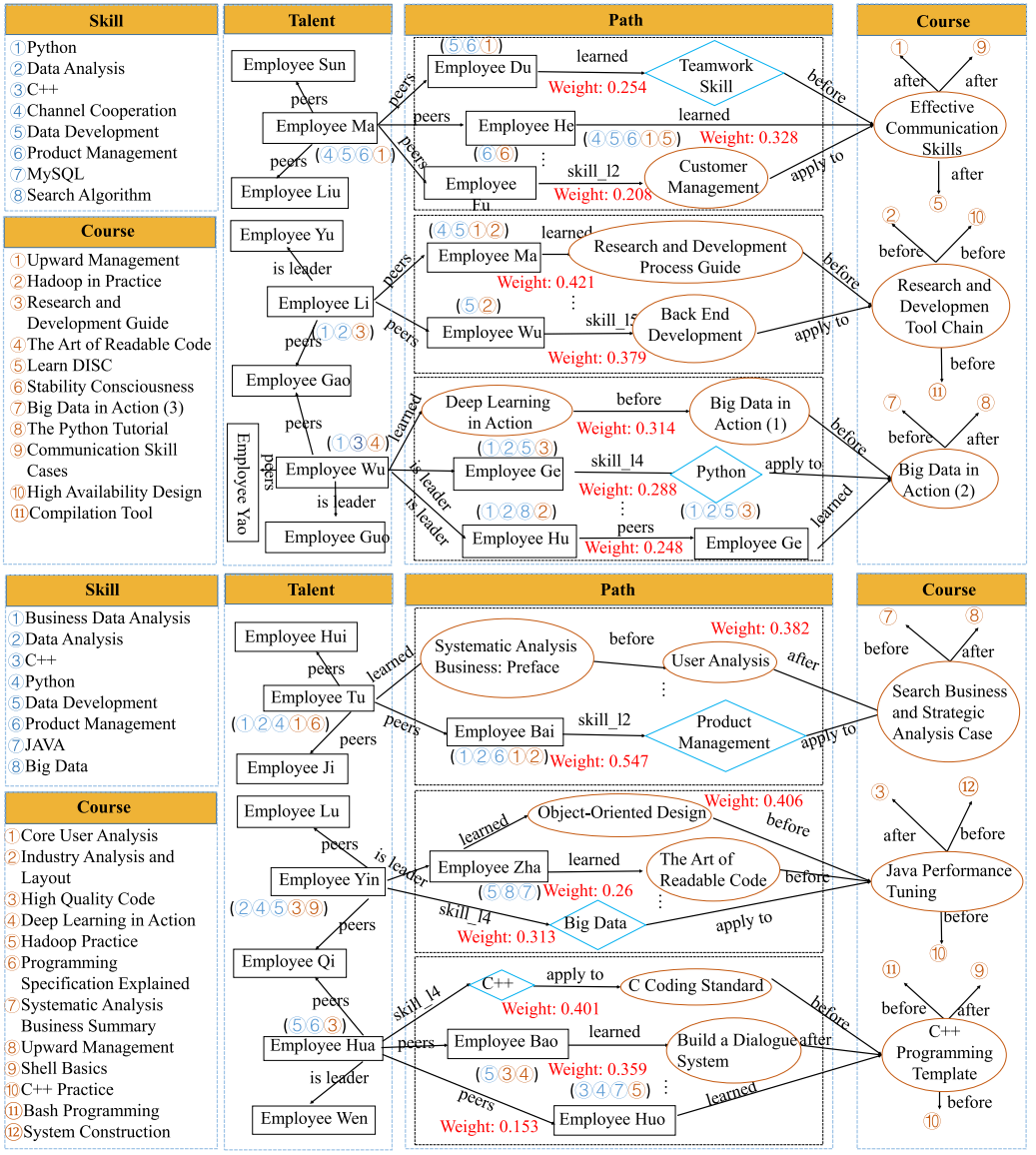

Fig. 7. Running examples to help illustrate the interpretability of learning motivation using the weights of meta-paths. In detail, we exhibit two examples, and the course entity is represented with an ellipse, the skill entity is represented with a diamond, and the talent entity is represented with a cube.

5.12 Explainability for Talent-Course Paths

CKGE not only can provide better recommendations but also can give a better interpretability for talent-course interactions. We can analyze the explainability by measuring the weight of different paths. The weight of path represents the degree of influence on the course clicked by talent. The larger value indicates greater influence. In Figure 7, we exhibit several talent-course pairs and the constructed meta-graphs, then acquire the weight for each path. Note that here we only show

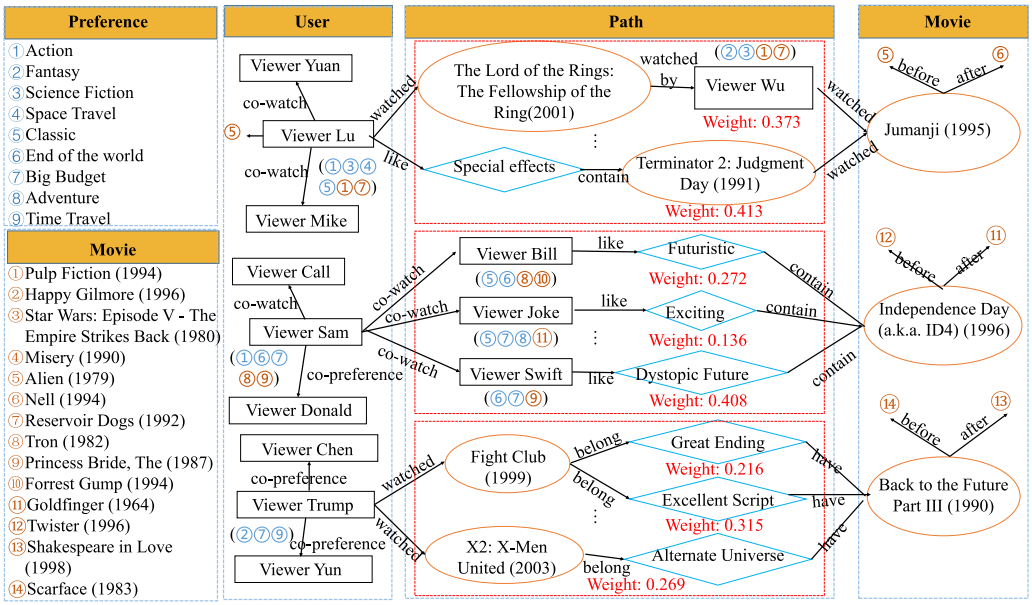

Fig. 8. Running examples to help illustrate the interpretability of Last.FM dataset. In detail, the user entity is represented with an ellipse, the preference entity is represented with a diamond, and the movie entity is represented with a cube.

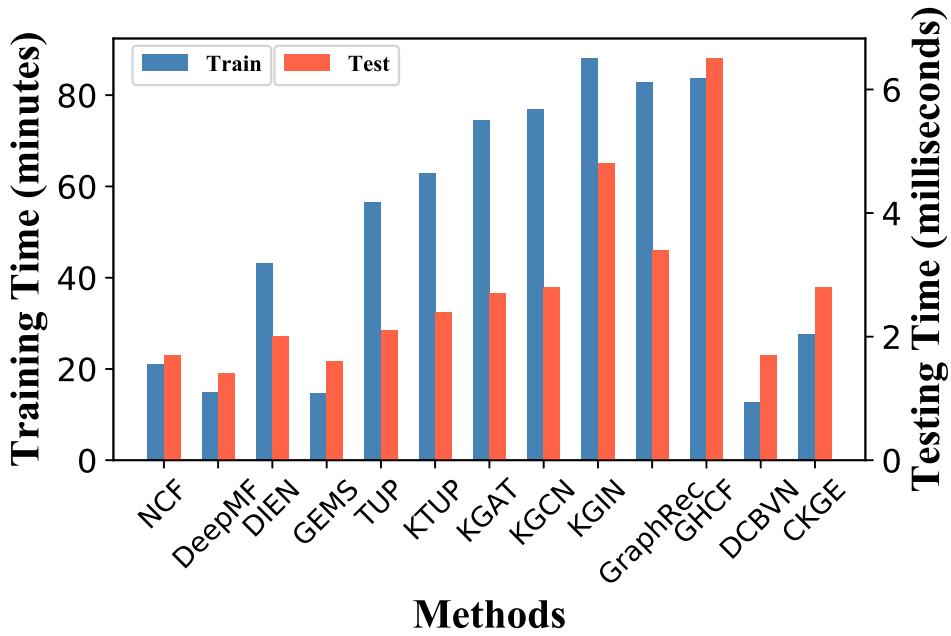

Fig. 9. The training and testing efficiency of CKGE and compared models.

the results of paths with higher weights due to page limitations. We utilize the first pair (i.e., "Employee Ma-Effectiveness Communication Skills") as an example. Several interesting findings are obtained, and we can find that the weight of "Employee Ma $\xrightarrow{Peers}$ Employee He $\xrightarrow{learned}$ Effective

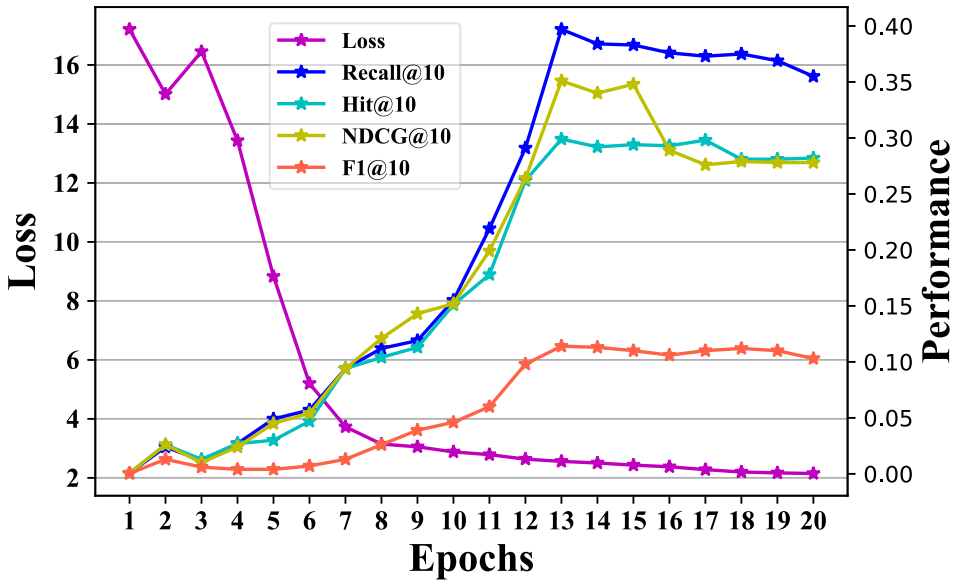

Fig. 10. An illustration of convergence versus performance of CKGE.

Communication Skills" is highest, for the reason that Employee Ma and Employee He have almost the same skills and course learning records, and are in a colleague type of relationship, so Employee Ma is very likely to choose the courses that Employee He has learned because of project needs or intra-group communication. In addition, the weight of path "Employee Ma $\overset{Peers}{\longrightarrow}$ Employee Du $\overset{learned}{\longrightarrow}$ Teamwork Skill $\overset{Before}{\longrightarrow}$ Effective Communication Skills" acquired is the second best, because the course "Teamwork Skill" is the prerequisite of the course "Effective Communication Skills," and Employee Du has more intersections with Employee Ma than Employee Fu, and therefore this path has a higher weight. Similar observations can be obtained for other examples.

Moreover, we conduct our proposed method on a public dataset to validate the generality. Figure 8 records the results, and we take the meta-graph of "Viewer Lu" as an example, where the weight assigned to the path "Viewer Lu $\overset{watched}{\longrightarrow}$ Lord of the Rings: The Fellowship of the Ring (2001) $\overset{watchedby}{\longrightarrow}$ Viewer Wu $\overset{watched}{\longrightarrow}$ Jumanji (1995)" is significantly higher than that of other paths. This is because "Viewer Lu" and "Viewer Wu" exhibit a strong correlation in both their movie preferences and viewing history, which makes "Viewer Lu" more likely to watch movies previously viewed by "Viewer Wu" based on their preferences and habits. Moreover, we can see from the path "Viewer Trump $\overset{watched}{\longrightarrow}$ X2: X-Men United (2003) $\overset{belong}{\longrightarrow}$ Alternate Universe $\overset{have}{\longrightarrow}$ Back to the Future Part III (1990)" that the preference "Alternate Universe" has the strongest correlation with both "X2: X-Men United (2003)" and "Back to the Future Part III (1990)," resulting in a higher weight for this path. Comparable findings can be observed in other examples.

5.13 Computational Efficiency

Furthermore, we exhibit the computational times of CKGE and typically compared baselines to evaluate the efficiency. As shown in Figure 9, we observe that the training time is 27 minutes, which is much shorter than comparison KG-based models and heterogeneous graph models. However, although the training time of CKGE is slightly higher than traditional models, CKGE outperforms

these methods. Moreover, after the training process, the average cost of each instance in the testing set is 2.8 ms. This clearly validates that our model can be effectively used in the real-world course recommendation system.

6 CONCLUSION

In this article, we developed an explainable training course recommender system with the consideration of different learning motivations of talents. To solve this problem, we proposed CKGE, a contextualized KG embedding approach that can learn effective representations of heterogenous KG entities by integrating both neighbor semantics and high-order connections as motivation-aware information. Specifically, we first constructed a meta-graph for each entity pair. Then, we developed a novel KG-based Transformer, which can serialize entities and paths in the meta-graph as a sequential input. Meanwhile, it promoted representations learning and motivation analysis through specially designed relational attention, structural encoding mechanisms, and local path mask prediction. As a result, CKGE not only could precisely predict the preference score of the talent-course pair but also could discriminate the saliencies of neighbors and meta-paths in characterizing corresponding preferences. Finally, extensive experiments on a real-world dataset demonstrated the effectiveness and interpretability of our approach.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* 17, 6 (2005), 734–749.

[2] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR* abs/1607.06450 (2016).

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*. 2787–2795.

[4] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2011. Second workshop on information heterogeneity and fusion in recommender systems (HetRec2011). In *Proceedings of the ACM Conference on Recommender Systems (RecSys'11)*.

[5] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *Proceedings of the World Wide Web Conference*. 151–161.

[6] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. 2019. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *Proceedings of the World Wide Web Conference*. 151–161.

[7] Chong Chen, Weizhi Ma, Min Zhang, Zhaowei Wang, Xiuqiang He, Chenyang Wang, Yiqun Liu, and Shaoping Ma. 2021. Graph heterogeneous multi-relational recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 3958–3966.

[8] Yun-Nung Chen, William Yang Wang, and Alexander I. Rudnicky. 2015. Jointly modeling inter-slot relations by random walk on knowledge graphs for unsupervised spoken language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 619–629.

[9] Louis V. DiBello, Louis A. Roussos, and William Stout. 2006. Review of cognitively diagnostic assessment and a summary of psychometric models. In *Handbook of Statistics*, C. R. Rao and S. Sinharay (Eds.). Vol. 26. North Holland, 979–1030.

[10] Yuntao Du, Xinjun Zhu, Lu Chen, Ziquan Fang, and Yunjun Gao. 2022. MetaKG: Meta-learning on knowledge graph for cold-start recommendation. *CoRR* abs/2202.03851 (2022).

[11] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *Proceedings of the World Wide Web Conference*. 417–426.

[12] Bairan Fu, Wenming Zhang, Guangneng Hu, Xinyu Dai, Shujian Huang, and Jiajun Chen. 2021. Dual side deep context-aware modulation for social recommendation. In *Proceedings of the Web Conference*. 2524–2534.

[13] Li Gao, Hong Yang, Jia Wu, Chuan Zhou, Weixue Lu, and Yue Hu. 2018. Recommendation with multi-source heterogeneous information. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 3378–3384.

[14] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *CoRR* abs/2003.00911 (2020).

[15] Zhenyu Han, Fengli Xu, Jinghan Shi, Yu Shang, Haorui Ma, Pan Hui, and Yong Li. 2020. Genetic meta-structure search for recommendation on heterogeneous information network. In *Proceedings of the ACM International Conference on Information and Knowledge Management*. 455–464.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[17] Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. 199–208.

[18] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the World Wide Web Conference*. 173–182.

[19] Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S. Yu. 2018. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1531–1540.

[20] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y. Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. 505–514.

[21] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2020. A survey on knowledge graphs: Representation, acquisition and applications. *CoRR* abs/2002.00388 (2020).

[22] Jim Kirkpatrick and Wendy Kayser Kirkpatrick. 2015. *An Introduction to the New World Kirkpatrick Model*. Kirkpatrick Partners.

[23] Walid Krichene and Steffen Rendle. 2021. On sampled metrics for item recommendation (extended abstract). In *Proceedings of the International Joint Conference on Artificial Intelligence*. 4784–4788.

[24] Yakun Li, Lei Hou, and Juanzi Li. 2023. Preference-aware graph attention networks for cross-domain recommendations with collaborative knowledge graph. *ACM Transactions on Information Systems* 41, 3 (2023), Article 80, 26 pages.

[25] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 2181–2187.

[26] Yong Liu, Susen Yang, Yonghui Xu, Chunyan Miao, Min Wu, and Juyong Zhang. 2023. Contextualized graph attention network for recommendation with item knowledge graph. *IEEE Transactions on Knowledge and Data Engineering* 35, 1 (2023), 181–195.

[27] Weizhi Ma, Min Zhang, Yue Cao, Woojeong Jin, Chenyang Wang, Yiqun Liu, Shaoping Ma, and Xiang Ren. 2019. Jointly learning explainable rules for recommendation with knowledge graph. In *Proceedings of the World Wide Web Conference*. 1210–1221.

[28] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online learning of social representations. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 701–710.

[29] Chuan Qin, Hengshu Zhu, Tong Xu, Chen Zhu, Chao Ma, Enhong Chen, and Hui Xiong. 2020. An enhanced neural network approach to person-job fit in talent recruitment. *ACM Transactions on Information Systems* 38, 2 (2020), 1–33.

[30] Yanru Qu, Ting Bai, Weinan Zhang, Jian-Yun Nie, and Jian Tang. 2019. An end-to-end neighborhood-based interaction model for knowledge-enhanced recommendation. *CoRR* abs/1908.04032 (2019).

[31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR* abs/1910.10683 (2020).

[32] Harshali Rane and Krishna Warhade. 2021. A survey on deep learning for intracranial hemorrhage detection. In *Proceedings of the International Conference on Emerging Smart Computing and Informatics*. IEEE, Los Alamitos, CA, 38–42.

[33] Brijesh Sivathanu and Rajasshrie Pillai. 2018. Smart HR 4.0—How Industry 4.0 is disrupting HR. *Human Resource Management International Digest* 26, 4 (2018), 7–11.

[34] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[35] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta path-based top-k similarity search in heterogeneous information networks. *Journal of VLDB Endowment* 4, 11 (2011), 992–1003.

[36] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge graph embedding by relational rotation in complex space. In *Proceedings of the International Conference on Learning Representations*.

[37] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Long-Kai Huang, and Chi Xu. 2018. Recurrent knowledge graph embedding for effective recommendation. In *Proceedings of the ACM Conference on Recommender Systems*. 297–305.

[38] Juntao Tan, Shijie Geng, Zuohui Fu, Yingqiang Ge, Shuyuan Xu, Yunqi Li, and Yongfeng Zhang. 2022. Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning. In *Proceedings of the World Wide Web Conference on World Wide Web*. 1018–1027.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.
[40] V. Vovk and R. Wang. 2012. Combining p-values via averaging. arXiv:1212.4966 (2012).
[41] Chao Wang, Hengshu Zhu, Peng Wang, Chen Zhu, Xi Zhang, Enhong Chen, and Hui Xiong. 2022. Personalized and explainable employee training course recommendations: A Bayesian variational approach. *ACM Transactions on Information Systems* 40, 4 (2022), Article 70, 32 pages.
[42] Chao Wang, Hengshu Zhu, Chen Zhu, Xi Zhang, Enhong Chen, and Hui Xiong. 2020. Personalized employee training course recommendation with career development awareness. In *Proceedings of the World Wide Web Conference*. 1648–1659.
[43] Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi Liu. 2018. SHINE: Signed heterogeneous information network embedding for sentiment link prediction. In *Proceedings of the ACM International Conference on Web Search and Data Mining*. 592–600.
[44] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep knowledge-aware network for news recommendation. In *Proceedings of the World Wide Web Conference on World Wide Web*. 1835–1844.
[45] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. 2019. Knowledge graph convolutional networks for recommender systems. In *Proceedings of the World Wide Web Conference*. 3307–3313.
[46] Quan Wang, Pingping Huang, Haifeng Wang, Songtai Dai, Wenbin Jiang, Jing Liu, Yajuan Lyu, Yong Zhu, and Hua Wu. 2019. CoKE: Contextualized knowledge graph embedding. *CoRR* abs/1911.02168 (2019).
[47] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge graph attention network for recommendation. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 950–958.
[48] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. 2021. Learning intents behind interactions with knowledge graph for recommendation. In *Proceedings of the World Wide Web Conference*. 878–887.
[49] Xiangmeng Wang, Qian Li, Dianer Yu, Zhichao Wang, Hongxu Chen, and Guandong Xu. 2022. MGPolicy: Meta graph enhanced off-policy learning for recommendations. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1369–1378.
[50] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 1112–1119.
[51] Xin Xin, Xiangnan He, Yongfeng Zhang, Yongdong Zhang, and Joemon M. Jose. 2019. Relational collaborative filtering: Modeling multiple item relations for recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. 125–134.
[52] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep matrix factorization models for recommender systems. In *Proceedings of the International Joint Conference on Artificial Intelligence*. 3203–3209.
[53] Rui Yan, Ran Le, Yang Song, Tao Zhang, Xiangliang Zhang, and Dongyan Zhao. 2019. Interview choice reveals your preference on the market: To improve job-resume matching through profiling memories. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery Data Mining*. 914–922.
[54] Mengyue Yang, Qingyang Li, Zhiwei (Tony) Qin, and Jieping Ye. 2020. Hierarchical adaptive contextual bandits for resource constraint based recommendation. In *Proceedings of the Web Conference*. 292–302.
[55] Yang Yang, Jia-Qi Yang, Ran Bao, De-Chuan Zhan, Hengshu Zhu, Xiaoru Gao, Hui Xiong, and Jian Yang. 2023. Corporate relative valuation using heterogeneous multi-modal graph neural network. *IEEE Transactions on Knowledge and Data Engineering* 35, 1 (2023), 211–224.
[56] Yang Yang, Jingshuai Zhang, Fan Gao, Xiaoru Gao, and Hengshu Zhu. 2022. DOMFN: A divergence-orientated multi-modal fusion network for resume assessment. In *Proceedings the 30th ACM International Conference on Multimedia*. 1612–1620.
[57] Qing Ye, Chang-Yu Hsieh, Ziyi Yang, Yu Kang, Jiming Chen, Dongsheng Cao, Shibo He, and Tingjun Hou. 2021. A unified drug–target interaction prediction framework based on knowledge graph and recommendation system. *Nature Communications* 12, 1 (2021), 6775.
[58] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform bad for graph representation? *CoRR* abs/2106.05234 (2021).
[59] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 974–983.
[60] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*. 283–292.

[61] Xiao Yu, Xiang Ren, Yizhou Sun, Bradley Sturt, Urvashi Khandelwal, Quanquan Gu, Brandon Norick, and Jiawei Han. 2013. Recommendation in heterogeneous information networks with implicit user feedback. In *Proceedings of the ACM Conference on Recommender Systems*. 347–350.

[62] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 353–362.

[63] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys* 52, 1 (2019), Article 5, 38 pages.

[64] Yongfeng Zhang and Xu Chen. 2020. *Explainable Recommendation: A Survey and New Perspectives*. Foundations and Trends in Information Retrieval. Now Publishers.

[65] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 5941–5948.

[66] Fan Zhu, Horace Ho-Shing Ip, Apple W. P. Fok, and Jiaheng Cao. 2007. PeRES: A personalized recommendation education system based on multi-agents & SCORM. In *Advances in Web Based Learning—ICWL 2007*. Lecture Notes in Computer Science, Vol. 4823. Springer, 31–42.

Received 9 June 2022; revised 2 March 2023; accepted 28 April 2023