Knowledge-Based Systems 233 (2021) 107484

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

# Knowledge-enhanced recommendation using item embedding and path attention

Yuan Lin, Bo Xu *, Jiaojiao Feng, Hongfei Lin, Kan Xu

*School of Computer Science and Technology, Dalian University of Technology, Dalian, China*

ARTICLE INFO

*Article history:*
Received 28 January 2021
Received in revised form 7 September 2021
Accepted 8 September 2021
Available online 14 September 2021

*Keywords:*
Knowledge graph
Item embedding
Preference propagation
Recommendation

ABSTRACT

Recommender systems have attracted widespread attention in various online applications. To effectively recommend the needed items of users, knowledge graphs have been introduced to provide rich and complementary information to infer user preferences in recommender systems. Existing efforts have explored user preferences through specific paths and item embedding in knowledge graphs. However, user preferences have hardly been fully captured because users and items are always separately modeled. To address this problem, we propose a model to represent items from a user's perspective that provides effective supplementary information. User preferences encoded in historically clicked items are propagated along links in the knowledge graph. We propose a gated attention unit to capture user preferences from specific types of paths. Based on the captured preference information through the knowledge graph and supplementary item information, we generate effective reasoning paths to infer the underlying rationale of user–item interactions using the sequential model. Through extensive experiments on real-world datasets, we demonstrate that the proposed model achieves significant improvements over the state-of-the-art solutions.



## 1. Introduction

Recommendation technologies aim to ease information overload on the internet and have been a hot research topic in industrial and academic fields. Effective recommendation algorithms are devised to make full use of user's historical behavioral information, such as purchase records, click information and evaluation information on e-commence websites, to accurately model user preferences.

Among various recommendation models, the collaborative filtering model is one of the most classic and effective algorithms [1]. With recent advances in deep learning, neural network-based recommendation models exhibit powerful capabilities in modeling user–item interactions. For example, the deep crossing model [2] converted sparse features into dense features through the embedding layer and obtained the final recommendation using transformed spliced feature vectors and a multilayer perceptron. The wide and deep model [3] combined a linear model that deals with interactions of sparse features and a deep model that captures potential feature interactions. Inspired by the wide and deep model, the DeepFM model [4] took the relationship between features into consideration using the FM model [5]. The DeepFM model not only extracted high-order features by a deep neural network but also paid attention to the interaction between high- and low-order features. Since the attention mechanism has been widely used in image classification [6] and machine translation [7], the AFM model [8] introduced the mechanism into the FM model [5] so that the attention network assigns distinct weights to the second-order interaction features. Although deep learning based models have achieved higher accuracy, their interpretability has much room for improvement in mathematical simulation and approximation.

To improve the interpretability of recommendation, many researchers seek to use other effective information in modeling user preferences, such as knowledge graphs. A knowledge graph (KG) not only contains various items but also involves the relationships between items, which makes it a potentially useful resource for recommendation. We illustrate an example of a knowledge graph in Fig. 1. From the example, we can see that edges in the knowledge graph can reveal the relationships between two items, such as the 'film->genre' edge that indicates film1 is a documentary. There are two main benefits of knowledge graphs in recommendation. One advantage is that knowledge graphs can provide abundant supplementary information on recommending items, which is beneficial for improving recommendation accuracy. Another benefit is that edges between items provide much more semantic information among items, which can enhance the interpretability of recommendations by leveraging reasoning along specific paths.

* Corresponding author.
*E-mail addresses:* zhlin@dlut.edu.cn (Y. Lin), xubo@dlut.edu.cn (B. Xu), jiaojiaofeng@mail.dlut.edu.cn (J. Feng), hflin@dlut.edu.cn (H. Lin), xukan@dlut.edu.cn (K. Xu).

https://doi.org/10.1016/j.knosys.2021.107484
0950-7051/

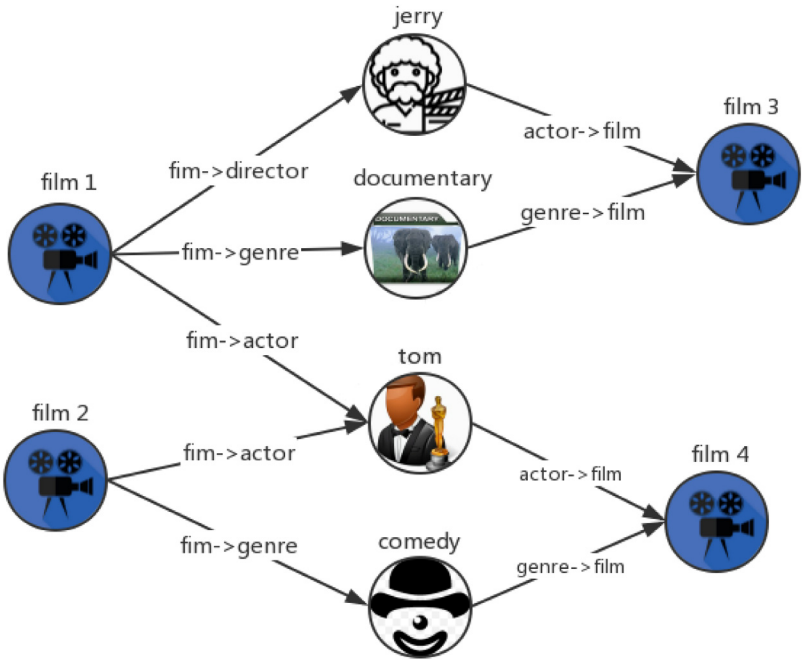


**Fig. 1.** An example of a knowledge graph.

The knowledge graph-based recommendation model has been proposed by using entity embedding for combined multiple links of knowledge graphs [9–17]. For example, the metapath-based model [11,18] learned effective relationship representations between users and items by walking on the empirically predefined metapath, which served as context and was combined with item–user embedding to improve recommendations. Another category of models apply graph embedding and entity embedding techniques, such as TransH [19] and TransR [20], to obtain effective item representations [21,22]. However, existing studies have been mostly focused on introducing supplementary information from knowledge graphs, partly ignoring the interpretability that can be generalized through knowledge graphs.

To address this problem, we propose a novel model CIEPA that uses the knowledge graph to combine item embedding and path attention for recommendation. Specifically, CIEPA predicts the user click probability for certain items based on his or her historical clicks. The core idea behind the CIEPA model lies in selective preference propagation. Selective preference propagation aims to model the hidden preference information in the interaction between users and items. The preference information propagates along links of the knowledge graph with significant user bias. We propose a user-level path attention mechanism to capture the user preferences. We generate global item vectors to encode an item co-occurrence matrix with respect to the same user. The global item vector provides hidden information that is not contained in the knowledge graph to ease the intrinsic incompleteness of the knowledge graph. When preference information propagates along links in the knowledge graph, the global item vector can filter out the spam information and acquire composite semantic information on the path.

The main contributions of this work are threefold: (1) We develop a path-level attention mechanism for explainable item recommendation. The mechanism provides reasonable explanations for user preferences toward different items and maintains high prediction accuracy. (2) We introduce global item vectors as complements to decompose the co-occurrence matrix of liked items to capture effective hidden information of items. (3) We conduct comprehensive experiments, and the results demonstrate the effectiveness of the proposed CIEPA model.

## 2. Related work

### 2.1. Item embedding

Currently, there are mainly two types of recommender systems, item-based recommendation and user-based recommendation. Item-based recommendation assumes that if two items are liked by the same user, the embedding of these two items tends to be nearer in the embedding space [23]. User-based recommendation [24–26] follows a similar idea in that if two users such as the same item, they may have similar interests in this kind of item. Therefore, how to obtain useful item embedding largely affects the effectiveness of recommender systems. To accurately represent items, many item embedding methods are proposed based on one-hot embedding [27], graph embedding [28] and word2vec [23,29]. Among existing embedding methods, word embedding and matrix factorization [1,5,30] are proven to be effective in item embedding that models user preferences [31]. For example, Liang [32] proposed a regularized multiembedding model that captures co-occurrence patterns of coliked and codisliked items, which substantially outperformed the state-of-the-art models in recommendation tasks.

### 2.2. Attention mechanism

The attention mechanism has been applied in recommender systems to assign distinguished weights on the input embedding of items. For example, the MCRec model developed deep neural networks with the coattention mechanism to leverage rich metapath-based contexts [11]. The SULM model analyzed the sentiment of user reviews to capture the most appealing aspect of items [33]. The deep interest network proposed by Zhou et al. [34] adopted an attentive network to model the different importance values of high-level features between the embedding layer and the concatenation layer. Wang et al. [9] explored path representations along the knowledge graph between users and items. They used an LSTM to model the paths connecting users and items to capture the preferences of users. Knowledge graphs have been used in recommendation tasks. Shi et al. [12] proposed a learning path recommendation model based on a multidimensional

knowledge graph framework for e-learning. The model generated personalized learning paths to improve e-learning experiences. Yang et al. [13] proposed the hierarchical attention graph convolutional network incorporating knowledge graph for explainable recommendation (HAGERec). The model captured users' potential preferences based on the high-order connectivity structure of a heterogeneous knowledge graph and provides in-depth explainability in recommendation. Wu et al. [14] proposed a novel exercise recommendation method. The method adopted recurrent neural networks (RNNs) and deep knowledge tracing (DKT) to capture the knowledge concept coverage for diversified novel recommendation. Zhu et al. [15] also addressed the problem of e-learning and proposed a new multiconstraint learning path recommendation algorithm based on a knowledge map for accurate path recommendation. Xie et al. [16] proposed an attentive metagraph embedding approach for item recommendation and demonstrated the model's superiority to baseline models. Ahmadian et al. [17] proposed using implicit relationships to enhance social recommendations based on a neighborhood improvement mechanism and obtained state-of-the-art recommendation performance. Alhamid et al. [35] proposed a novel personalized recommendation model that employed context for modeling user preferences and achieved higher accuracy over state-of-the-art algorithms. Rawashdeh et al. [36] addressed the problem of tag recommendation in folksonomy using a graph-based approach that modeled folksonomy as a weighted undirected tripartite graph and effectively improved recommendation performance.

In this work, we propose the dual hierarchical attention in the proposed model, including the item attention and the path attention. The item attention is computed based on the distribution of item triples in each hop to capture the most significant interest of users. The path attention captures user preferences for choosing a specific path. Item attention helps explain which genre appeals to a certain user, while the path attention can distinguish the importance of different lines of user interest.

## 3. Methodology

### 3.1. Overall framework

In this section, we introduce the proposed CIAPE framework in detail. The overall framework of our model is illustrated in Fig. 2. The proposed model treats the item embedding and user embedding as inputs and outputs the click probability of items. For a specific user, his or her historical clicked items are taken as the seed node in knowledge graph. The item information of the seed node is then propagated to its linked items through several hops in the graph. $Hop^k_{user}$ denotes $k$th hops from user's seed node. For each hop, specific preference embedding of the user $u$ is computed based on item embedding and path attention. To fully capture the item information, we introduce a global vector model that takes the co-occurrence matrix of items as inputs and outputs the item global vectors. The item global vectors are then concatenated with the preference embedding of the first hop. The concatenated embedding is fed into a long short-term memory network (LSTM) as the last inputs, in which the last hop's preference embedding is set as the first input of the LSTM. Finally, we predict the click probability using the combined preference embedding by the LSTM and the refined item embedding that concatenates the corresponding entity embedding and global embedding.

### 3.2. Modeling the preference propagation in a knowledge graph

#### 3.2.1. Ripple set

Knowledge graphs have various entities and edges. As illustrated in Fig. 1, entity types can be generally divided into two categories: item nodes and item attribute nodes. For example, the item attribute nodes of a movie-related graph include nodes representing actors, directors, genres, countries and languages. Inspired by the idea of the RippleNet model [37], we consider that node information related to items can be propagated in knowledge graph. Since the interaction between an item and a user encodes the user preference, we believe that user preferences can also be propagated along the paths in a knowledge graph. Taking the original interactions, such as clicks of items by users, as the seed node, item entities of the $k$th hop from the seed item nodes can be defined as follows.

$$N^k_{user} = \left\{t|(h, r, t) \in g, h \in N^{k-1}_{user}\right\} k = 1, 2, \ldots, H \quad (1)$$

where $g$ represents the knowledge graph in the form of triples ($h$, $r$, $t$). $h$, $r$ and $t$ represent the head node, relationship and tail node in one triple of knowledge, respectively. $H$ represents the number of hops. When $k = 0$, the set of items is the clicked items by the user.

The propagation process is analogous to a ripple. The list of nodes that interacts with a user is viewed as multiple raindrops falling into calm water that can cause a water ripple net. The propagation process of the ripple can be represented as multiple hops from the center to the distance, each hop containing a different set of entities. The $k$th hop ripple set of user $u$ is defined as the set of knowledge triples activated by the items in $N^{k-1}_{user}$.

$$Hop^k_{user} = \left\{(h, r, t)\,|(h, r, t) \in g, h \in N^{k-1}_{user}\right\} \quad k = 1, 2, \ldots, H \quad (2)$$

where $H$ represents the number of hops. As the propagation distance increases, the useful information decreases gradually, and noisy information continuously increases. In the wave propagation process generated by the original seed node, if the breadth (the number of triples in each hop) and the depth of propagation (the number of hops) are unlimited without taking some filtering measures, it not only leads to an increase of the computational burden but also reduces the effectiveness of the model. In the next subsections, we introduce our solution to limit the breadth and depth in ripple propagation.

#### 3.2.2. Preference propagation

In a knowledge graph, each item node is associated with a $d$-dimensional item embedding $V \in R^d$. The preference information of user $u$ for item v in the $k$th hop, denoted as $P^k_{u\to v}$, can be obtained using a mapping function between the embedding of item v and $Hop^k_{user}$ of user u. For each triple $\left(h^k_i, r^k_i, t^k_i\right)$ in $Hop^k_{user}$, the similar probability between $h^k_i$ and item v under relationship $r^k_i$ can be computed as follows.

$$p^k_i = softmax(attention\left(r^u_i\right) v^T r^k_i h^k_i) \quad (3)$$

where $r^k_i \in R^{d\times d}$ and $h^k_i \in R^d$ are the embeddings of the relation and the head item, respectively. $i$ represents the $i$th triple in the $k$th hop. The relation in the same categories in different hops has the same vectors. $attention\left(r^u_i\right)$ represents user $u$'s preference for the type of $r^u_i$, which reflects how much attention that user $u$ gives to different item attributes.

Noticeably, the knowledge graph, as a heterogeneous network, involves more than one type of edge. In heterogeneous network embedding, the semantic information of each edge type is different, and the incompatibility of semantic information has attracted extensive attention from scholars [38,39]. Taking the knowledge graph of movies as an example, when the head node of a triple

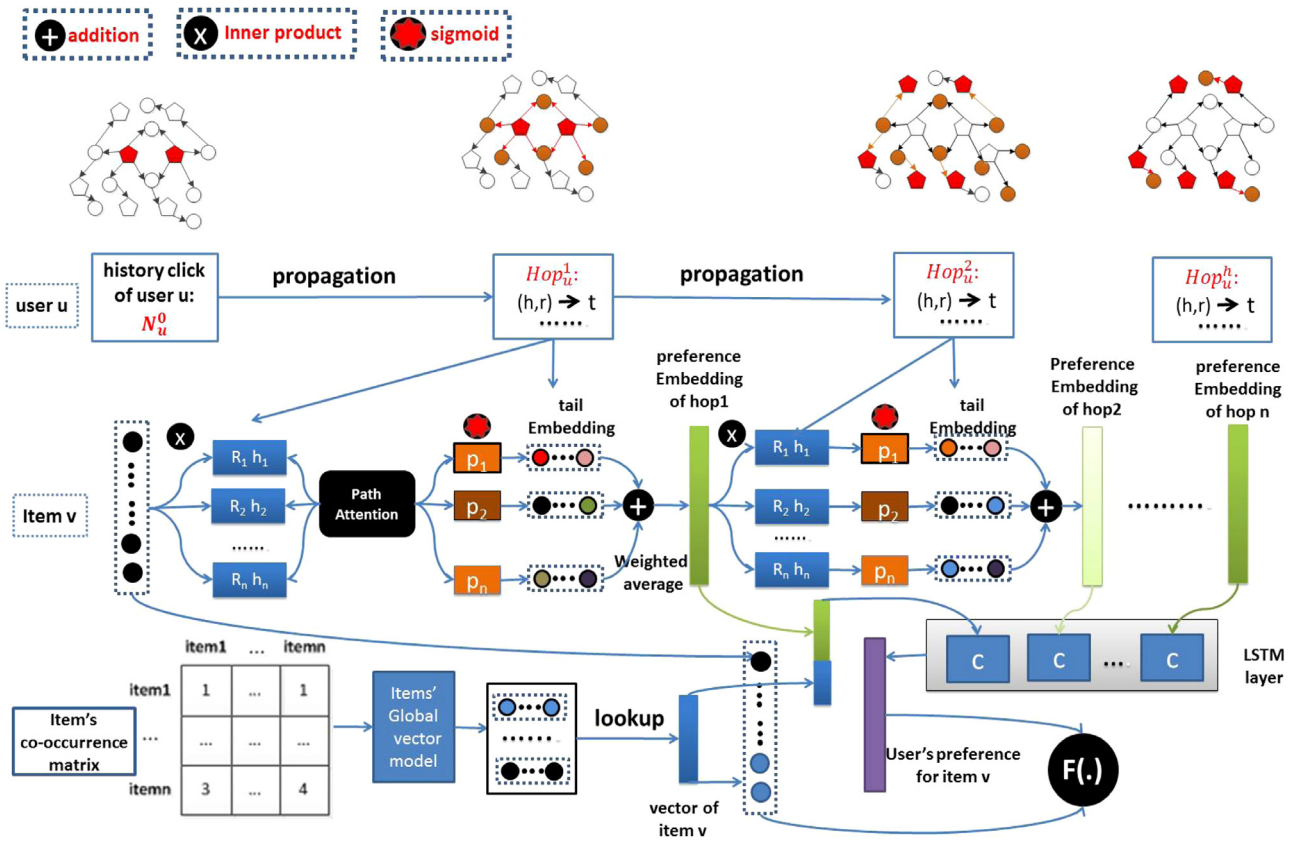


**Fig. 2.** The framework of CIAPE. This figure illustrates the propagation process activated by the user's click history, where F(.) represents the function to compute the prediction probability. The framework treats the item embedding and user embedding as inputs and outputs the click probability of items. For a specific user click (the top-left corner), his or her historical clicked items are taken as the seed node for modeling user preference. The item information of the seed node is then propagated to its linked items through several hops in the graph. $Hop^k_{user}$ denotes the kth hop from user's seed node. For each hop, specific preference embedding of the user $u$ is computed based on item embedding and path attention. Moreover, a global vector model is used to take the co-occurrence matrix of items as inputs and outputs the item global vectors. The item global vectors are then concatenated with the preference embedding of the first hop. The concatenated embedding is fed into a long short-term memory network (LSTM) as the last inputs. Finally, the click probability is predicted using the combined preference embedding by the LSTM and the refined item embedding that concatenates the corresponding entity embedding and global embedding.

represents a movie entity, the types of its directed linked edge include 'film.genre', 'film.director', 'film.actor', and 'film.language'. The corresponding tail node represents the entity of the movie genre, person (directors and actors are of the same type, and there is an intersection between them; some directors might also have the role of an actor) and language types. Edges of different types with incompatible semantic information can be used to generate distinctive representation vectors. Since different weight matrices are assigned to different edges, the similarity between the head node and the item node in different relation semantic spaces can be calculated.

### 3.2.3. Path attention

Inspired by the attention-driven factor model [40] that takes the attention-driven integration layer to capture users' attention distributions on different item features, we propose a path attention mechanism to address user preference relations in a knowledge graph. We formulate this idea as follows.

$$a^u_{ri} = softmax(w_a u_i) \tag{4}$$

where $w_a \in R^{(n+1)\times k}$ denotes the attention mapping matrix. $attention\,(r^u) = \{a^u_{r0}, a^u_{r1} \ldots\ldots a^u_{rn}\}$ indicates the attention distribution of the $i$th user. $a^u_{ri}$ denotes user u's preferences on edge type $r^i$. $n$ is the number of edge types. To avoid the overfitting problem of parameters, mapping matrix $w_a$ is shared by all users.

To explain the motivation of path attention, we illustrate some data statistics in Fig. 3, which includes the number of specific edges in the knowledge graph of the movie and the book and the number of corresponding tail nodes. From this figure, we can find that the distribution of the edge types and the distribution of the attribute nodes are very unbalanced in the knowledge graph.

When k is an odd number, triple (h,r,t) in $Hop^k_{user}$ represents the head of an item (for the book dataset, the item is a book, while for the movie dataset, the item is a ma movie), a, and the tail is the item's attributes. The number of triples in $Hop^k_{user}$ is limited by the computational burden; in another words, the propagation breadth of each layer needs to be constrained. For the $Hop^1_{user}$ of the user, different users can exhibit different choice tendencies for different item characteristics. A certain user may give more attention to the genre of an item than to a movie's actor, but another user may be more concerned about the film's actor. Therefore, we take the path attention into consideration to capture the different preferences of relations in the knowledge graph.

### 3.2.4. Combination of user preferences in multiple hops

After obtaining the similarity probabilities defined in Eq. (3), we consider how to combine the user preferences among multiple hops. We define the vector $o^k_{user}$ that reflects the user preference in the $k$th hop. For the triple $(h, r, t)$ (i) in $Hop^k_{user}$, $o^k_{user}$ can be computed as the sum of tail item embeddings in $Hop^k_{user}$ weighted by the corresponding similarity probabilities, which is computed as follows.

$$o^k_{user} = \sum_{(h_i, r_i, t_i) \in Hop^k_{user}} p_i t_i \tag{5}$$

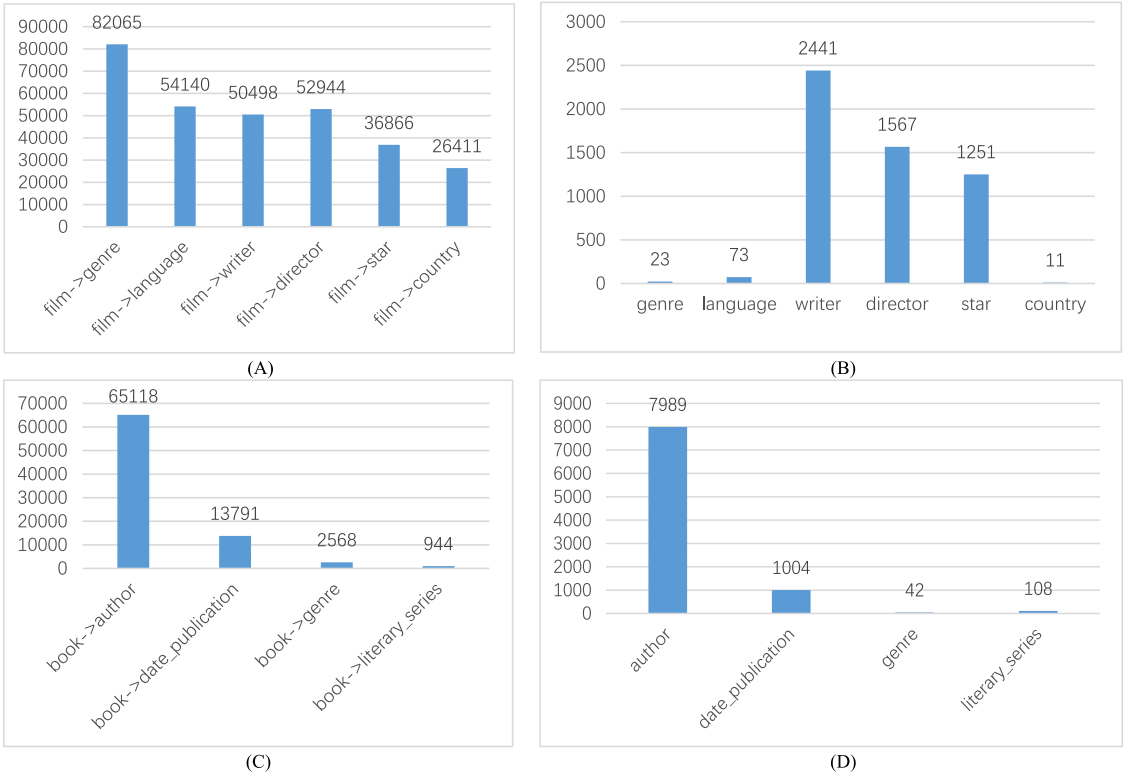


**Fig. 3.** The number of edge types and the number of corresponding tail node entities on two datasets, movie and book. Figure (A) depicts the statistics about edge types and their corresponding numbers in the movie dataset. "film->language" denotes the linked edge starting from the head of film types to the tail of ff film's language type. Figure (B) shows the number of corresponding tail node entities of the edge with a specific type for the movie dataset. Figures (C) and (D) refer to the book dataset.

where $t_i \in R^d$ is the embedding of tail $t_i$. $o^k_{user}$ represents a user's $k$-order preference toward item $v$. The user preference information for the item is obtained by modeling the similarity between the item and other items that are historically clicked by the same user.

In the process of user preferences propagating along the path, multiple layers of preference information vectors $o^1_{user}$, $o^2_{user}$ … $o^k_{user}$ can be obtained by repeating the above steps. Effective information in $o^k_{user}$ gradually decreases as the number of hops increases. Therefore, there is a sequence of preference embeddings that can be modeled using a sequence-based model, such as the time-series model. To this end, we propose using a sequential model LSTM to generate a single representation of the user preference for item $v$ by encoding the preference information among multiple hops. Since the LSTM has the ability to memorize the long-term dependency in a sequence and its forget gate can filter some noisy information, we adopt it in our model.

The LSTM model outputs the hidden state vector $h_l$ based on a subsequence $[o^k_{user}, o^{k-1}_{user}\ldots\ldots o^0_{user}]$. With the increase of $k$, the preference vector in the $k$th hop contains more noisy information and is at a farther distance from the user's clicked items. Therefore, the $o^k_{user}$ encodes more useful information when the hop is nearer to the final preference hidden state. Consequently, $h_l$ and $o^{k-1}_{user}$ are used to learn the hidden state of the next path, which is defined via the following equations:

$$\check{c}_l = \tanh(w_c[o^{k-l+1}_{user}, h_{l-1}] + b_c) \quad (6)$$

$$f_l = \sigma(w_f[o^{k-l+1}_{user}, h_{l-1}] + b_f) \quad (7)$$

$$i_l = \sigma(w_i[o^{k-l+1}_{user}, h_{l-1}] + b_i) \quad (8)$$

$$o_l = \sigma(w_o[o^{k-l+1}_{user}, w_h] + b_o) \quad (9)$$

$$c_l = f_l \odot c_{l-1} + i_l \odot \check{c}_l \quad (10)$$

where $i_l$, $o_l$ and $f_l$ represent the input gate, output gate and forget gate, respectively. $w_c$, $w_f$, $w_i$, $w_o \in R^{dh\times(dh+d)}$ are the mapping matrices and $b_c$, $b_f$, $b_i$ and $b_o$ are bias vectors. $\check{c}_l$ and $c_l \in R^{dh}$ denote the cell state and the control module for input information, respectively. When deterring the current state of the unit, $\check{c}_l$ can control how input information affects the current state's change. $dh$ is the dimension of the hidden state. $\sigma$ is the sigmoid activation function. $\odot$ is the elementwise product of two vectors. The combination of the memory unit and the forget gate can make it possible to memorize effective preference information and filter noise information when the depth of propagation increases. $o^0_{user}$ and the vector of item v are concatenated to form the last input of the LSTM.

### 3.2.5. Global vectors of items

Item embedding provides useful information to improve knowledge graph-based recommendation accuracy in recommendation systems. Knowledge graphs contain informative nodes and edges, and each of them plays different roles to construct item embedding. To filter the noisy information in constructing item embedding, we take some filter strategies to capture more useful information.

Specifically, we introduce the global vectors for word representation by GloVe [41], which is widely used in the field of natural language processing. As a language model, based on the overall statistics of a term's frequency, it can represent a word as a vector that captures the semantic properties between words. Semantic similarity between two words can be obtained by calculating the Euclidean distance or cosine similarity of two vectors. We then introduce how to construct the GloVe vectors for items in a recommendation. GloVe vectors for items are constructed based on the original corpus. First, a co-occurrence matrix is constructed. Each element in the matrix represents the number

of times that the word $i$ and the context word $j$ co-occur in a context window with a particular size. The GloVe model takes into account the distance between words in the sliding window. A decaying weighting decay=1/d is used to calculate the weight, which means the farther the distance, the smaller the weight is.

Inspired by this idea, we consider that from the user's point of view, two item co-occurrences in positive item lists of a user can indicate some common features between them. The higher co-occurrence time means that they share more common features, which may not be included in the knowledge graph. Therefore, we define the co-occurrence numbers for two items as the frequency that they are liked by the same user. In other words, the co-occurrence number is equal to the number of users who such as two items. To smooth the data, the number of occurrences can be divided by the maximum number of co-occurrences. Data smoothing aims to limit the obtained co-occurrence-based metric in the interval of [0, 1], which helps alleviate calculation bias caused by differences in the magnitudes of values. We divided the raw number of co-occurrences by the maximum number of co-occurrences because the maximum number of co-occurrences is the largest value of all the co-occurrence numbers. In this way, the resulting value will be in the interval of [0, 1] and will facilitate the next calculations.

$$X_{\text{item,item2}} = co_{clicktime}/\max(co\_clickTime) \tag{11}$$

where $X_{\text{item,item2}}$ represents co-occurrence times of item1 and item2. To accumulate all the co-occurrences of a certain item and other items, we can sum all the $X_{\text{item,item2}}$ with respect to the item. We formulate this idea as follows.

$$X_{item1} = \sum_{item2} X_{\text{item1,item2}} \tag{12}$$

where $X_{item1}$ represents the sum of co-occurrence times of *item1* and other items. Eq. (12) is used to compute all the co-occurrences of *item1* and other items. Namely, the equation reflects the popularity of a certain item among all the items. If another item such as is i*tem2* co-occurs frequently with the *item1*, the user preferences between *item1* and *item2* can be modeled. Therefore, we formulate the probability of a user who likes both item1 and item2 as follows.

$$p_{item1,item2} = X_{\text{item1,item2}}/X_{item1} \tag{13}$$

where $p_{item1,item2}$ represents the probability of *item2* appearing in co-occurrence items of *item1*. Eq. (13) measures the probability of co-occurrences of *item1* and *item2*, which encodes user preferences between different items and can be used for global item representations.

The core idea of the GloVe model is that for word $i$ and word $j$, if word $k$ has a closer semantic relationship with the word $i$ compared with the semantic similarity between word $k$ and word $j$, the probability $p_{\text{i,k}} > p_{\text{j,k}}$, where the probability $p_{\text{i,k}}$ indicates the probability that two items ($i$ and $k$) are liked by the same user. Since the vector representation of the item is associated with the co-occurrence matrix from the view of user preference, the obtained vector representation of items will contain the user preference information and item information.

$$F\left(v_{item1}, v_{item3}\right)/F\left(v_{item2}, v_{item3}\right) = p_{item1,item3}/p_{item2,item3} \tag{14}$$

where $w_{item1}w_{item2}$ and $w_{item3}$ are vector representations of item1, item2 and item3, respectively. $F$ represents the operation function between vectors, where we adopt the dot product operation.

For an item, each of its co-occurring items should have different weights in determining its vector representation. The model's loss function is defined as the weighted mean square loss. $f$ is a nondecreasing function, which satisfies $f(0) = 0$ and has its maximum peak.

$$\text{loss} = \sum_{item1,item2=1}^{n} f(X_{item1,item2})\left(w_{item1}^{T} w_{item2} + b_{item1} + b_{item2} - \log X_{\text{item1,item2}}\right)^2 \tag{15}$$

Eq. (15) is based on the weighted least squares regression model used by GloVe [41]. $w_{item1}^{T}w_{item2} + b_{item1} + b_{item2} - \log X_{item1,item2}$ is a drastic simplification of the cost function over Eq. (14). $f(X_{item1,item2})$ is a weighting function introduced into the cost function. GloVe has been proven to be effective in different natural language processing tasks. Readers can refer to reference [41] for more details about the approximation and derivation of the equations.

### 3.3. Learning algorithm

Following related works [9,37,42], we view the recommendation task as a binary classification problem. A user–item pair is labeled as 1 when the user has an interaction with the item; otherwise, a pair is labeled as 0. The first part of our loss function is the cross-entropy loss function [37].

$$l_1 = \sum_{(u,v)\in d} \left(\sigma\left(\underset{u^T}{\rightarrow}\underset{v}{\rightarrow}\right)\right)^{y^{uv}} \cdot \left(1 - \sigma\left(\underset{u^T}{\rightarrow}\underset{v}{\rightarrow}\right)\right)^{1-y^{uv}} \tag{16}$$

where $(u, v)$ represents any user–item pair in a given dataset $d$. $\overrightarrow{v}$ represents the item embedding vector, and $\overrightarrow{u}$ represents the user embedding vector with respect to the item $v$. $\sigma$ is the linear activation sigmoid function. $y^{uv}$ represents the user–item pair label of $(u, v)$. The cross-entropy loss function in Eq. (16) measures the recommendation loss of predicting the interaction label of each pair of item and user. In the process of optimizing the model, we intend to maximize the posterior probability of model parameters based on the observations of knowledge graph $g$ and implicit feedback $Y$. Model parameters include the embeddings of all entities, relations between items and users, and the path attention.

For each triple $(h, r, t) \in$ knowledge graph, there are various methods to obtain the embedding of entities and relations to capture the node and structural information. The trans series model, such as transE and transG, regards the representations of nodes and relationship in the knowledge graph as a machine translation problem. Network embedding focuses on semantic matching. The knowledge graph contains various nodes of different types and edges with incompatible semantic information. To learn information from the multirelational knowledge graph, we use a tensor factorization model that takes the inherent structure of relational data into account [43].

$$l_2 = \sum_{(h,r,t)\in g} \left(I_{h,r,t} - h^T r t\right)^2 \tag{17}$$

where h, t $\in R^d$ denote the latent representations of the head entity and the tail entity, respectively, r $\in R^{d\times d}$ denotes the latent representations of the edge between the head entity and the tail entity. Each entity has a unique latent representation. Each edge with different types has different vector representations. If (h,r,(t) $\in$g, $I_{h,r,t}$ equals 1; otherwise, $I_{h,r,t}$ equals 0.

Another concern is user preferences on different edges. When user preferences propagate from $N^0_{\text{user}}$ to $N^1_{\text{user}}$, some edges encode more user preferences than others due to the unbalanced distribution of edges. For example, there are various types of edges in the movie dataset: 'film → genre' connects a movie and its genre, 'film → director' connects a movie and its director, and 'film → language' connects a movie and its language. According

to the statistics of the data, it can be observed that a different user has different attentions on the edge. This attention of users on edges can be associated with the distribution of the tail node of the edge.

$$tail^u_{ri} = \{t|(h, r, t) \in Hop^1_u \text{ and } ri \in type(r)\} \tag{18}$$

where $tail^u_{ri}$ denotes the tail entity connected by the relation of type $ri$ (such as film.genre) in $Hop^1_u$.

$$t\_type_{ri} = set\{t|(h, r, t) \in Hop^1_u \quad (u \in users) \text{ and } ri \in type\,(r)\} \tag{19}$$

where $t\_type_{ri}$ denotes the set of the tail types connected by the relation of type $ri$ in $Hop^1_{users}$. In this set, each element is unique and an element with the same name will be saved only one. We take a specific user into count. The equation can be formulated as follows.

$$t\_type^u_{ri} = set\{t|(h, r, t) \in Hop^1_u \text{ and } ri \in type(r)\} \tag{20}$$

where $t\_type^u_{ri}$ denotes the set of tail types connected by the relation of the type $ri$ $ii$ in $Hop^1_u$ for the user u. Based on Eqs. (18) and (20), we can compute user preferences for a certain type of edge as follows.

$$weight^u_{ri} = \sum_{t^j \in t^u_{type\ ri}} \frac{|t^j|}{|tail^u_{ri}|} * \log \frac{|t^j|}{|tail^u_{ri}|} \tag{21}$$

$$wa^u_r = softmax(weight^u_{r0,} \ldots weight^u_{ri} \ldots weight^u_{rn}) \tag{22}$$

where $weight^u_{ri}$ denotes the user preference for the edge of type $ri$ based on information entropy. $|t^i|$ is the number of the $i$th tail entity. To explain the motivation for this, we provide a simple example. For two users u1 and u2, if u1 historically interacted with movies directed by 10 directors, while u2's number is 20, we would consider that u1 may care more about the film director attribute than u2. From another perspective, unbalanced interaction may reflect user preferences because if the number of historically interacted directors of u1 and u2 is the same and equal to 10 but the distribution is unbalanced, such as [2, 2, 2, 2, 2] and [1, 1, 1, 2, 5], it is observed that they have different preferences. A more balanced distribution implies that the user has less preference information for this attribute. *u2* focuses more on the attribute of director than *u1*. To obtain user preferences for edges, we define the third part of the loss function as:

$$l_3 = \frac{1}{|users|} \sum_{u \in users} \sum_{ri \in R} \left(attention\left(r^u\right) - wa^u_r\right)^2 \tag{23}$$

where $|users|$ represents the number of users. $wa^u_{ri}$ is defined in Eq. (22). We then combine the three parts of the loss functions for model learning.

## 4. Experiments

### 4.1. Experimental settings

To demonstrate the effectiveness of the proposed model, we use two publicly available datasets MovieLens-1M [10] and DBbook2014. These two datasets are widely used in related tasks for movie recommendation and book recommendation. The knowledge graph of these two datasets has been constructed by Wang et al. [37]. We use the constructed knowledge graph to learn the knowledge-enhanced models. In the experiments section, we would such as to answer the following four research questions:

**RQ1**: What is the performance of the proposed CIEPA model? Does it outperform other state-of-the-art models? (See Section 4.2)

**RQ2**: How does the number of hops affect the preference propagation in our model? (See Section 4.3)

**RQ3**: How does the size of the ripple set in each hop affect the model performance? (See Section 4.4)

**RQ4**: What is the influence of embedding dimensionality and path attention weights for the proposed model? (See Section 4.5)

We examine the model effectiveness to answer the above four research questions and provide in-depth analysis and discussion of our model at the end of this section.

We compare the proposed model with the following state-of-the-art baselines. We choose five strong baseline models. All the compared models follow the same experimental settings for fair comparison. The baseline models are introduced as follows.

- **The NFM model** [44] developed a bi-interaction layer to combine second-order features based on a factorization machine. Tail entities of the first hop represent categorical features of items and users, and therefore, the tail entity embedding in the first hop is concatenated with the user ID and item ID in the NFM model.
- **The CKE** model [45] introduced structural information, textual information and visual information into collaborative filtering. In our comparison, the CKE model is combined with collaborative filtered structural information of the knowledge graph.
- **The PER model** [21] treats the knowledge graph as a heterogeneous information network and extracts metapath-based features to represent the connectivity between users and items. In our comparison, we use all item–attribute–item features for the PER model (e.g., "movie–director–movie").
- **The RippleNet model** [37] views the propagation of user preference information on the knowledge graph as ripples and randomly constructs the triples in ripple-like hops.
- **The DeepFM model** [4] consists of two parts that share the same inputs: the neural network part and the factorization machine part. The neural network part extracts the low-level features and the factorization machine part extracts the high-level features.

Hyperparameters of these models are tuned by optimizing AUC on the validation set. To control the depth of propagation, we tune the number of hops as 2 for the MovieLens-1M and DBbook2014 datasets. For each dataset, the ratio of the training, evaluation, and test set is set to be 3:1:1, which follows common settings in related works. We split the dataset 10 times into the training, validation and test set and report the average performance over 10 test sets. The reported results are from all the test runs. The model is evaluated by the click-through rate (CTR) prediction. The learned model outputs the predicted click probability for each test set. We use area under the curve (AUC) and accuracy (ACC) as the evaluation metrics in our experiments.

### 4.2. Overall experimental results

To answer **RQ1**, we first compare our model with the baseline models. The experimental results of all the compared models are evaluated by the click-through rate prediction and shown in Table 1. In the table, CIEPA-ATT and CIEPA are two versions of the proposed models. CIEPA denotes the proposed model with path attention, and CIEPA-ATT denotes the proposed model without path attention. A two-tailed paired $t$ test ($p \leq 0.05$) is conducted to demonstrate the significance of performance improvement. Significant improvement over the best-performed baseline (RippleNet) is indicated with an asterisk (*).

**Table 1**
Performance comparison by AUC and Accuracy in CTR prediction.

| MODEL | DBbook2014 | | MovieLens-1M | |
|---|---|---|---|---|
| | AUC | ACC | AUC | ACC |
| NFM [44] | 0.7095 | 0.6804 | 0.8668 | 0.7932 |
| RippleNet [37] | 0.7290 | 0.6620 | 0.9210 | 0.8441 |
| DeepFM [4] | 0.6759 | 0.6296 | 0.8774 | 0.8041 |
| CKE [45] | 0.6740 | 0.6350 | 0.7960 | 0.7390 |
| PER [21] | 0.6230 | 0.5880 | 0.7120 | 0.6670 |
| CIEPA-ATT | **0.7331*** | **0.6880*** | 0.9216 | 0.8478 |
| CIEPA | 0.7015 | 0.6537 | **0.9254*** | **0.8496*** |

From the results, we observe that the deep learning-based methods yield better results on both datasets. NFM performs better than DeepFM on the DBbook dataset in terms of AUC and ACC, while DeepFM outperforms NFM in the MovieLens dataset. PER achieved relatively low performance compared with other methods. One possible reason is that the performance of PER highly depends on the predefined metapaths, which may result in worse results due to the lack of flexibility. In contrast, the proposed model produces better results than the baseline models on both datasets. For the two versions of the proposed models, CIEPA-ATT achieves better results for book recommendation, while CIEPA achieves better results for movie recommendation. The results indicate that path attention can reduce the recommendation performance on the book dataset. Therefore, we further examine the reasons for this finding.

We divide the users into different groups based on the distribution of the number of interacted items. The distribution of users is illustrated in Fig. 4. From the figure, we can see that the movie dataset contains 442 users out of 6050 users who rated less than 5 items, accounting for approximately 7.3% of the total users, while the book datasets contains 11,589 users who rated less than 5 items, accounting for approximately 89.7% of the total users. The unbalanced distribution of the number of user ratings on the book dataset limits the capability of our model in capturing useful information in the knowledge graph. Specifically, in our model, the preference propagating process treats the user's historically clicked items as seed nodes. User preferences propagate from the seed item nodes to other item attribute nodes. Compared to the movie dataset, the book dataset produces a more sparse knowledge graph. Namely, book seed nodes have fewer outgoing links than movie seed nodes. As a result, user preferences for certain paths cannot be easily obtained. Therefore, path attention cannot easily capture path preferences on the book dataset, although the CIEPA model yields better performance than the baseline models.

### 4.3. Influence of the number of hops

To answer **RQ2**, in this section, we examine the influence of the number of hops in our model. Intuitively, with the number of hops increasing, the preference information would continuously decrease from the seed nodes to the target nodes. Our model adopts LSTM to encode user preference information and filter noisy information. We compare our model with RippleNet on the movie dataset and report the comparison performance by AUC in Table 2. From Table 2, we observe that RippleNet and CIPEA have a similar trend with the number of hops increasing. When we set the number of hops as 2, both models can obtain most preference information and yield the best AUC value. In comparison, CIEPA can constantly capture more effective information than RippleNet by switching the number of hops. Therefore, we set the number of hops as 2 in our experiment.

**Table 2**
The results of AUC with different hop numbers.

| Model/#Hop | #Hop=1 | # Hop =2 | # Hop =3 | # Hop =4 |
|---|---|---|---|---|
| RippleNet | 0.9160 | 0.9210 | 0.9150 | 0.9180 |
| CIPEA | 0.9212 | 0.9254 | 0.9214 | 0.9217 |

**Table 3**
The performance by AUC with different sizes of the ripple set.

| Size of ripple set | 16 | 32 | 64 | 96 |
|---|---|---|---|---|
| CIPEA/movie | 0.9213 | 0.9234 | 0.9214 | 0.9254 |
| CIPEA-ATT/book | 0.7092 | 0.7317 | 0.7323 | 0.7331 |

**Table 4**
The results with different dimensionalities of GloVe item embedding on the movie dataset.

| GloVe dim. | AUC | ACC |
|---|---|---|
| 25 | 0.9084 | 0.8318 |
| 50 | 0.9168 | 0.8424 |
| 75 | 0.9169 | 0.8418 |
| 100 | 0.9112 | 0.8346 |
| 200 | 0.9216 | 0.8478 |
| 300 | **0.9254** | **0.8496** |
| 400 | 0.9208 | 0.8467 |

**Table 5**
The results with different dimensionalities of GloVe item embedding on the book dataset.

| GloVe dim. | AUC | ACC |
|---|---|---|
| 25 | 0.7292 | 0.6783 |
| 50 | 0.7302 | 0.6824 |
| 75 | 0.7258 | 0.6844 |
| 100 | 0.7317 | 0.6790 |
| 200 | **0.7331** | **0.6880** |
| 300 | 0.7245 | 0.6856 |
| 400 | 0.7303 | 0.6815 |

### 4.4. Influence of the size of the ripple set in each hop

To answer **RQ3**, we vary the size of a user's ripple set in each hop to further investigate the effectiveness of our model. The results on the movie datasets are shown in Table 3. For both datasets, AUC increases as the size of the ripple set in each hop increases. When the size of the ripple set is equal to 96, the performance reaches its peak. On the movie dataset, path attention effectively captures user preferences; namely, the key path with higher attention conveys more preference information. When the ripple size increases, less weight is assigned to the noisy information. We observe a similar trend on the book dataset. The performance increases with the increasing size of the ripple set and tends to flatten when the size of the ripple set is larger than 96.

### 4.5. Influence of the embedding dimensionality and the path attention weight

To answer **RQ4**, in this section, we investigate the influence of the weight of the path attention loss and the dimensionality of the GloVe embedding. The influence of the path attention weight on the movie dataset is shown in Fig. 5. From the figure, we observe that CIEPA achieves better results when the path attention weight is set to be 1e-5. One possible explanation for this finding is that a large weight could exaggerate the path-preference information, while a small value can assign appropriate attention on paths. Path attention is not fit for the book dataset as discussed above.

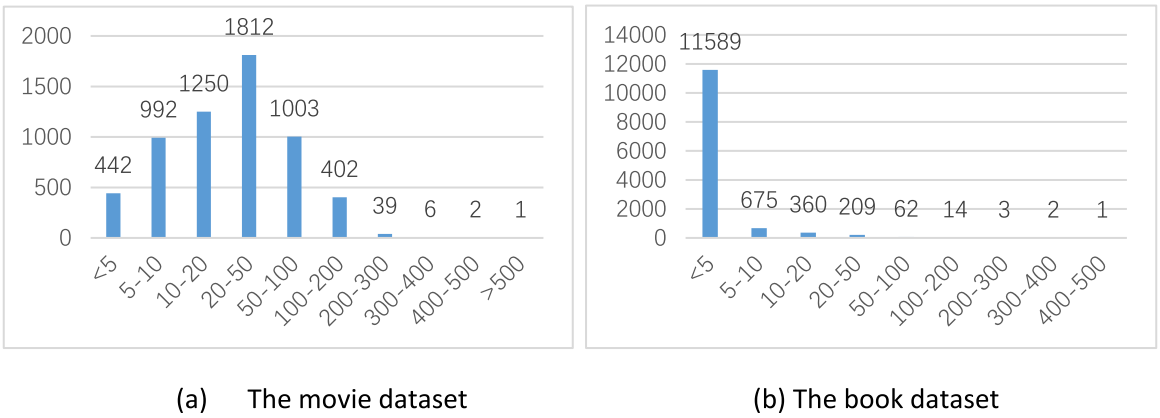


**Fig. 4.** User group distribution based on the number of rating items.

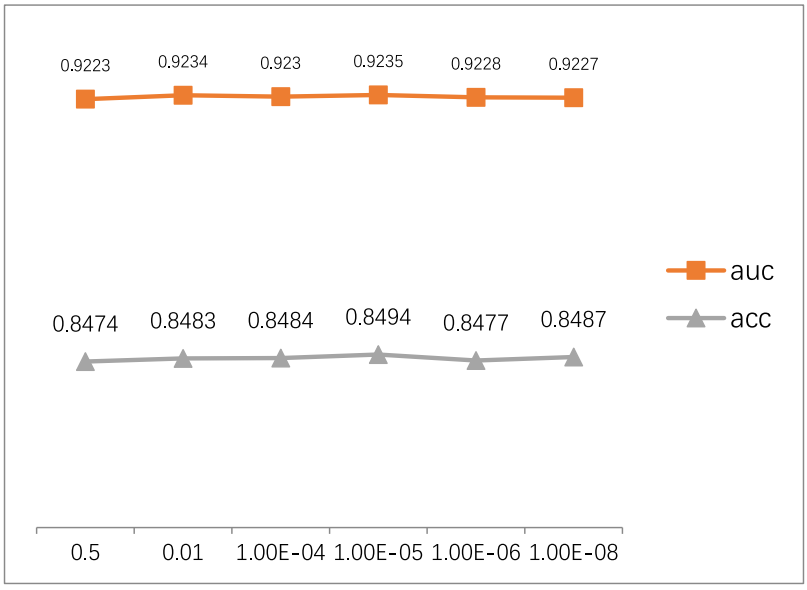


**Fig. 5.** Training weight of path attention loss.

We then examine the influence of dimensionality of GloVe item embedding on both datasets. The results are reported in Tables 4 and 5. From the tables, we switch the dimensionality of GloVe item embedding to obtain the best-performing parameter. On the movie dataset, when the dimensionality is set to be 300, we can obtain the best performance. On the book dataset, when the dimensionality is set to be 200, we can obtain the best performance.

### 4.6. Discussions

In this section, we provide further discussion and analysis on the proposed model. Based on the experimental results, we find that our model outperforms other state-of-the-art models on both datasets, which demonstrates the effectiveness of the proposed framework in capturing user preferences. Specifically, on the DBbook2014 dataset, the proposed CIEPA-ATT model significantly outperforms the best-performing baseline model RippleNet with an ACC improvement of 3.93%, although the path attention cannot yield good performance due to data sparsity. On the MovieLens-1M dataset, the proposed CIEPA model significantly increases the performance of RippleNet with an ACC improvement of 0.65%. The experimental results indicate the proposed model can improve the recommendation performance based on item embedding and path attention within multiple hops.

According to the experimental results, we attribute the performance improvement to three aspects. First, we introduce the knowledge graph as an informative resource to enhance item recommendation. Knowledge graphs provide extra item information and the relationship of items and attributes for accurate preference modeling. According to the experimental results in Tables 2 and 3, our model can constantly capture more effective information than other baseline models by switching the number of hops and ripple size in the knowledge graph.

Second, we propose the path attention mechanism to capture the propagation of user preferences within the knowledge graph based on the idea of ripple nets. Path attention assigns appropriate weights on different knowledge paths to leverage complete user interests based on historical behaviors. According to the experimental results in Fig. 4, the path attention weight can be optimized during model learning and continuously reduce the training loss to capture more user preferences.

Third, we introduce the global item embedding as an effective complement to embed useful item information in the final recommendation model. The global item embedding decomposes the co-occurrence matrix of coliked items to capture effective hidden information of items. According to the experimental results in

Tables 4 and 5, we observe that the global item embedding can constantly provide effective information to enhance our model and improve the model effectiveness.

Benefiting from these aspects, our model achieves better recommendation performance in the experiments. In addition, our method still has some limitations, which could be further optimized in the future. As shown in the experimental results, path attention is highly dependent on abundant user ratings. If user ratings are sparse, such as those in the book dataset, path attention may not work well. This problem can be jointly considered with the cold start issue in recommender systems. To tackle this problem, other useful information can be employed to relieve the negative influence, such as user social networks and codislike information with respect to certain items.

In regard to the computational performance of our model, compared with RippleNet [37], extra computational cost mainly comes from the path attention mechanism. In our model, path attention is proposed to model the unbalanced distribution of edge types and attribute nodes in the knowledge graph. We compute path attention based on the number of hops and the size of the ripple set. Therefore, the computational significance of path attention is linearly proportional to the number of hops and the size of the ripple set, which will not bring more computational overhead than other knowledge graph-based models, such as RippleNet. Namely, the computational performance of our model is comparable to other state-of-the-art models.

## 5. Conclusion and future work

In this paper, we propose a novel knowledge-enhanced recommendation framework CIEPA. The CIEPA framework incorporates the knowledge graph into a coliked co-occurrence matrix with path-level attention. Specifically, the learned recommendation model combines the embedding-based and path-based knowledge-aware information from two aspects. On the one hand, our model introduces item global vectors to filter noisy information for refined user preferences. On the other hand, we propose path attention to guide preference propagation and infer underlying user–item interaction for accurate item recommendation. We conduct experiments on two publicly available datasets, and experimental results demonstrate that our model significantly outperforms other state-of-the-art baseline models. Since our framework is general, other useful information can also be integrated into the CIEPA framework in the future. For example, contents or description information of items can be used to enrich the item representation, which may ease the cold start problem in recommending items to new users. Moreover, negative samples indicating that a user dislikes certain items can also be employed to better model user preferences in knowledge-enhanced recommendation.

## CRediT authorship contribution statement

**Yuan Lin:** Writing - review & editing, Investigation. **Bo Xu:** Conceptualization, Methodology, Writing - original draft. **Jiaojiao Feng:** Experiments, Coding. **Hongfei Lin:** Formal analysis, Funding acquisition. **Kan Xu:** Writing - review & editing, Validation.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This work is partially supported by grant from the Natural Science Foundation of China (No. 61976036, 62006034, 62076046), the National Social Science Foundation, China (20BTQ074), Natural Science Foundation of Liaoning Province, China (2021-BS-067).

## References

[1] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer (8) (2009) 30–37.
[2] Y. Shan, T.R. Hoens, J. Jiao, et al., Deep crossing: Web-scale modeling without manually crafted combinatorial features, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 255–262.
[3] H.T. Cheng, L. Koc, J. Harmsen, et al., Wide & deep learning for recommender systems, in: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, ACM, 2016, pp. 7–10.
[4] H. Guo, R. Tang, Y. Ye, et al., DeepFM: A factorization-machine based neural network for CTR prediction, 2017.
[5] S. Rendle, Factorization machines, in: 2010 IEEE International Conference on Data Mining, IEEE, 2010, pp. 995–1000.
[6] V. Mnih, N. Heess, A. Graves, Recurrent models of visual attention, in: Advances in Neural Information Processing Systems, 2014, pp. 2204–2212.
[7] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, in: 3rd International Conference on Learning Representations, ICLR 2015.
[8] J. Xiao, H. Ye, X. He, et al., Attentional factorization machines: Learning the weight of feature interactions via attention networks, in: International Joint Conference on Artificial Intelligence, IJCAI, 2017, pp. 3119–3125.
[9] X. Wang, D. Wang, C. Xu, et al., Explainable reasoning over knowledge graphs for recommendation, in: The Thirty-Third AAAI Conference on Artificial Intelligence, 2019, pp. 5329–5336.
[10] B. Hu, C. Shi, W.X. Zhao, et al., Leveraging meta-path based context for top-n recommendation with a neural co-attention model, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, 2018, pp. 1531–1540.
[11] Y. Dong, N.V. Chawla, A. Swami, metapath2vec: Scalable representation learning for heterogeneous networks, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2017, pp. 135–144.
[12] D. Shi, T. Wang, H. Xing, et al., A learning path recommendation model based on a multidimensional knowledge graph framework for e-learning, Knowl.-Based Syst. 195 (2020) 105618.
[13] Z. Yang, S. Dong, HAGERec: Hierarchical attention graph convolutional network incorporating knowledge graph for explainable recommendation, Knowl.-Based Syst. 204 (2020) 106194.
[14] Z. Wu, M. Li, Y. Tang, et al., Exercise recommendation based on knowledge concept prediction, Knowl.-Based Syst. 210 (2020) 106481.
[15] H. Zhu, F. Tian, K. Wu, et al., A multi-constraint learning path recommendation algorithm based on knowledge map, Knowl.-Based Syst. 143 (MAR.1) (2018) 102–114.
[16] F. Xie, A. Zheng, L. Chen, et al., Attentive meta-graph embedding for item recommendation in heterogeneous information networks - ScienceDirect, Knowl.-Based Syst. (2020) 211.
[17] S. Ahmadian, N. Joorabloo, M. Jalili, et al., A social recommender system based on reliable implicit relationships, Knowl.-Based Syst. (2019) 105371.
[18] B. Hu, C. Shi, W.X. Zhao, et al., Leveraging meta-path based context for top-n recommendation with a neural co-attention model, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, 2018, pp. 1531–1540.
[19] Z. Wang, J. Zhang, J. Feng, et al., Knowledge graph embedding by translating on hyperplanes, in: Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014.
[20] Y. Lin, Z. Liu, M. Sun, et al., Learning entity and relation embeddings for knowledge graph completion, in: Twenty-ninth AAAI conference on artificial intelligence, 2015.
[21] X. Yu, X. Ren, Y. Sun, et al., Personalized entity recommendation: A heterogeneous information network approach, in: Proceedings of the 7th ACM International Conference on Web Search and Data Mining, ACM, 2014, pp. 283–292.
[22] J. Huang, W.X. Zhao, H. Dou, et al., Improving sequential recommendation with knowledge-enhanced memory networks, in: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, ACM, 2018, pp. 505–514.

[23] O. Barkan, N. Koenigstein, Item2vec: neural item embedding for collaborative filtering, in: 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing, MLSP, IEEE, 2016, pp. 1–6.
[24] Maksims Volkovs, Guang Wei Yu, Effective Latent Models for Binary Feedback in Recommender Systems, in: SIGIR 2015, pp. 313–322.
[25] J. Yu, M. Gao, J. Li, et al., Adaptive implicit friends identification over heterogeneous network for social recommendation, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, ACM, 2018, pp. 357–366.
[26] O. Levy, Y. Goldberg, Neural word embedding as implicit matrix factorization, in: Advances in Neural Information Processing Systems, 2014, pp. 2177–2185.
[27] Xiang Wang, Xiangnan He, Meng Wang, FuliFeng, Tat-Seng Chua, Neural graph collaborative filtering, in: SIGIR'19, Paris, France, July 21–25, 2019.
[28] J. Wang, P. Huang, H. Zhao, et al., Billion-scale commodity embedding for e-commerce recommendation in alibaba, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, 2018, pp. 839–848.
[29] H. Caselles-Dupré, F. Lesaint, J. Royo-Letelier, Word2vec applied to recommendation: Hyperparameters matter, in: Proceedings of the 12th ACM Conference on Recommender Systems, ACM, 2018, pp. 352–356.
[30] Y. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets, in: ICDM, Vol. 8, 2008, pp. 263–272.
[31] O. Levy, Y. Goldberg, Neural word embedding as implicit matrix factorization, in: Advances in Neural Information Processing Systems, 2014, pp. 2177–2185.
[32] D. Liang, J. Altosaar, L. Charlin, et al., Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence, in: Proceedings of the 10th ACM Conference on Recommender Systems, ACM, 2016, pp. 59–66.
[33] D. Wang, G. Xu, S. Deng, Music recommendation via heterogeneous information graph embedding, in: 2017 International Joint Conference on Neural Networks, IJCNN, IEEE, 2017, pp. 596–603.
[34] G. Zhou, X. Zhu, C. Song, et al., Deep interest network for click-through rate prediction, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, 2018, pp. 1059–1068.
[35] M.F. Alhamid, M. Rawashdeh, H. Dong, M.A. Hossain, A.E. Saddik, Exploring latent preferences for context-aware personalized recommendation systems, IEEE Trans. Hum. Mach. Syst. 46 (4), 615–623.
[36] M. Rawashdeh, H.N. Kim, J.M. Alja'am, et al., Folksonomy link prediction based on a tripartite graph for tag recommendation, J. Intell. Inf. Syst. 40, 307–325.
[37] H. Wang, F. Zhang, J. Wang, et al., RippleNet: Propagating user preferences on the knowledge graph for recommender systems, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, ACM, 2018, pp. 417–426.
[38] Y. Shi, H. Gui, Q. Zhu, et al., Aspem: Embedding learning by aspects in heterogeneous information networks, in: Proceedings of the 2018 SIAM International Conference on Data Mining, Society for Industrial and Applied Mathematics, 2018, pp. 144–152.
[39] Y. Shi, Q. Zhu, F. Guo, et al., Easing embedding learning by comprehensive transcription of heterogeneous information networks, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '18, London, United Kingdom (2018.08.19–2018.08.23), ACM Press, 2018, pp. 2190–2199.
[40] J. Chen, F. Zhuang, X. Hong, et al., Attention-driven factor model for explainable personalized recommendation, in: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, ACM, 2018, pp. 909–912.
[41] J. Pennington, R. Socher, C. Manning, GloVe: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing, EMNLP, 2014, pp. 1532-1543.
[42] X. He, L. Liao, H. Zhang, et al., Neural collaborative filtering, in: Proceedings of the 26th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2017, pp. 173–182.
[43] M. Nickel, V. Tresp, H.P. Kriegel, A three-way model for collective learning on multi-relational data, in: ICML.11, 2011, pp. 809–816.
[44] X. He, T.S. Chua, Neural factorization machines for sparse predictive analytics, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2017, pp. 355–364.
[45] T. Tran, K. Lee, Y. Liao, et al., Regularizing matrix factorization with user and item embeddings for recommendation, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, ACM, 2018, pp. 687–696.