scientific reports

Check for updates

OPEN

Enhanced knowledge graph recommendation algorithm based on multi-level contrastive learning

Zhang Rong[1]✉, Liu Yuan[2] & Li Yang[1]

Integrating the Knowledge Graphs (KGs) into recommendation systems enhances personalization and accuracy. However, the long-tail distribution of knowledge graphs often leads to data sparsity, which limits the effectiveness in practical applications. To address this challenge, this study proposes a knowledge-aware recommendation algorithm framework that incorporates multi-level contrastive learning. This framework enhances the Collaborative Knowledge Graph (CKG) through a random edge dropout method, which constructs feature representations at three levels: user-user interactions, item-item interactions and user-item interactions. A dynamic attention mechanism is employed in the Graph Attention Networks (GAT) for modeling the KG. Combined with the nonlinear transformation and Momentum Contrast (Moco) strategy for contrastive learning, it can effectively extract high-quality feature information. Additionally, multi-level contrastive learning, as an auxiliary self-supervised task, is jointly trained with the primary supervised task, which further enhances recommendation performance. Experimental results on the MovieLens and Amazon-books datasets demonstrate that this framework effectively improves the performance of knowledge graph-based recommendations, addresses the issue of data sparsity, and outperforms other baseline models across multiple evaluation metrics.

Keywords Recommendation system, Knowledge graph, Knowledge-aware recommendation, Contrastive learning, Graph Neural Network (GNN)

Recently, researchers introduced KG into a recommendation system to improve performance. These graphs involve rich semantic and structural information, which can contribute to achieving more accurate, diversified and interpretable recommendations. This method is called the knowledge graph-aware recommendation (KGR), which strengthens the representation of users and items by encoding additional item-level semantic associations[1]. Given the natural ability of GNNs to encode collaborative signals and their proficiency in extracting multi-hop relationships, utilizing GNNs for KGR has currently become one of the mainstream modeling frameworks for recommendation systems, as shown in the reference[2,3]. These models spread the characteristics of items in the KG and use a GNN to model the KG to assist the recommendation system. The performance of the KGR model depends largely on the quality of KG. However, the KG in the real world is often incomplete. A serious long-tail distribution problem exists, that is, a few nodes have a large number of edges. It can be imagined that several celebrities on social networks have more followers than most ordinary users. This imbalance[4] causes the existence of a large number of super-nodes, which brings challenges including sparse supervision signals and affects the accurate capture of user preferences by the model.

In contrastive learning[5], by comparing different data representation forms, it is possible to know more robust and distinguishing features and make remarkable achievements in computer vision and other fields. This result proves its potential in handling sparse and unbalanced data. Contrastive learning is introduced into the recommendation system. By comparing various representations of items, the model can dig out deeper semantic associations, thus effectively focusing on long tail nodes. This graph contrastive learning method is mainly based on two core components: data enhancement and contrastive learning. Specifically, data enhancement creates multiple views of nodes by generating multiple variants for each node. Contrastive learning focuses on analyzing and learning the consistency between these different views to promote more robust feature learning. At present, many researchers focus on the field of graph data enhancement. For instance, references[6,7]proposed the structure-based graph data enhancement technology. Reference[8] studied the feature-based method. These methods have been proven to significantly improve the model performance. Although the model based on graph

[1]School of Internet of Things Engineering, Jiangsu Vocational College of Information Technology, Wuxi 214153, China. [2]School of Artificial Intelligence and Computer, Jiangnan University, WuXi 214122, China. ✉email: 95291188@qq.com

contrastive learning has made some achievements in the recommendation system, it faces some challenges as follows:

1) Most studies focus more on data enhancement technology and less on improving contrastive learning itself, which limits the potential of the model.

2) At present, contrastive learning usually only compares different views of the same node, which ignores the relationship between users and items. This practice fails to fully explore the relationship between users and items, which may reduce the accuracy and personalization of recommendations.

To solve the above problems, this paper proposes a recommendation algorithm framework, namely the Multi-level-Knowledge Dynamic Graph Attention Network Momentum Contrast (ML-KDGATMoco). This framework integrates KG and user-item interaction into a unified graph structure (as shown in Fig. 1), which is handled in the following ways:

1) Combining the three types of nodes in the graph structure, such as users, items and knowledge and their relationships, a three-level contrastive learning task is constructed by taking random edge dropout technology as an auxiliary self-supervision to strengthen the model performance.
2) GAT is used for deep learning of graph structure, and the representation of KG is further optimized through dynamic attention mechanism and nonlinear transformation.
3) In the contrastive learning stage, momentum encoder and updating strategy are introduced to construct high-quality negative samples. Hence, the contrastive learning effect is improved.
4) Multi-level contrastive learning is taken as an auxiliary task. It is jointly trained with the primary supervised task of knowledge-aware recommendation to effectively solve the problem of data sparseness.

The main contributions of this paper are as follows:

1) A three-level contrastive learning strategy is designed, including user-user, item-item and user-interactive item contrastive learning. This contrastive learning is realized by using nonlinear transformation and negative sample construction strategy of momentum encoder.
2) A dynamic attention mechanism is introduced into the GNN architecture, which effectively improves the ability of KG modeling and enhances the overall performance of the recommendation system.
3) Taking contrastive learning as an auxiliary self-supervised task, it is trained jointly with a knowledge-aware recommendation task. This strategy promotes the representation learning of the model. Therefore, performance in the recommendation task is improved.

Related work

Recommendation based on KG

Integrating KGs into the recommendation system as auxiliary information can improve the accuracy of recommendations and provide additional interpretability for the results. There are three main methods in the research of Knowledge Graph Embedding (KGE) recommendation systems. To be specific, the first is the solid embedding method. For example, the embedding technology that the CFKG model[9]relies on is TransE[10], while that for the CKE model[11]is TransR[12]. These methods integrate the representation of KG by generating item embedding and then combine it with the interactive data between users and items. The second is the path-based

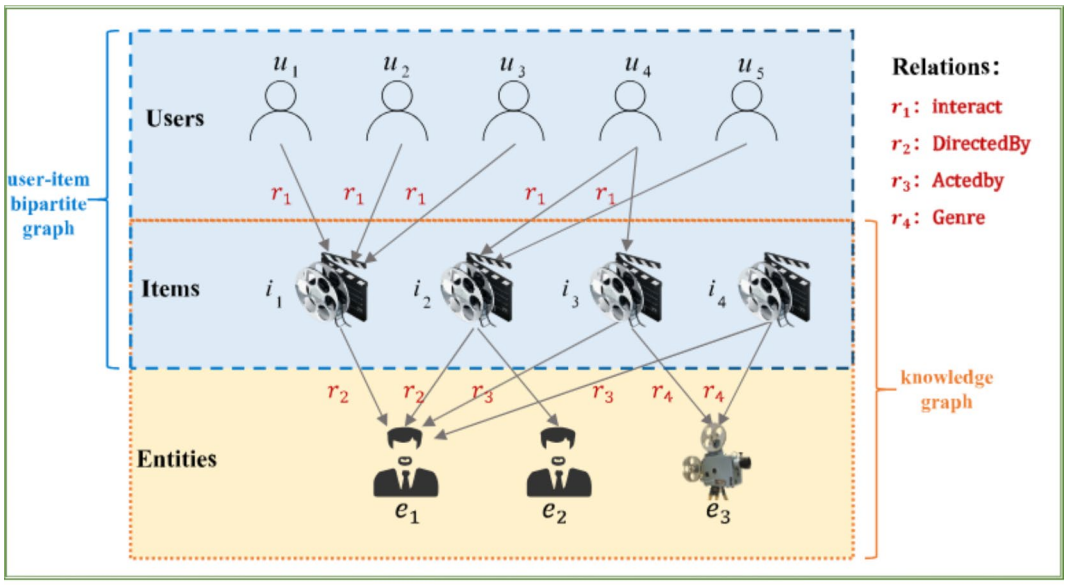

Fig. 1. Collaborative KG.

method. The KG can form a heterogeneous graph with the data of user-interactive items in the recommendation system. Therefore, researchers propose a Meta-path method[13]to deeply mine this kind of heterogeneous graph. For instance, Xiting Wang et al[14]. utilized multi-level reinforcement learning for recommendations, and S. Guo et al[15]. applied similar techniques for disease diagnosis, which expanded path-based strategies beyond standard recommendations. The third one adopts the GNN technology. These technologies propagate information recursively between multi-hop nodes, considering the long-range relationship structure. For example, the knowledge graph attention (KGAT) network model[3]combines the KG and user-items graph and realizes end-to-end modeling of the high-order relationship between users and items by constructing the KGAT network. Building upon these GNN-based approaches, recent sophisticated models like and TGCL4SR[16]M3KGR[17] have emerged, focusing on enhancing the temporal and multi-modal aspects of knowledge graphs to improve sequential recommendation systems.

The integration of KGs' semantics with GNNs' insights sets a path for developing recommendation systems that are accurate and aligned with users' detailed preferences.

Application of contrastive learning in recommendation systems

In traditional recommendation systems, methods including collaborative filtering[18]and matrix decomposition[19] exhibit robust performance with ample data. However, their efficacy witnesses notable diminishes when there is data sparsity.

Contrastive learning as a novel paradigm has garnered considerable attention for its exceptional ability to learn under few-shot conditions[20,21]. This approach enhances the distinction of feature representations by minimizing the distance between positive pairs and maximizing that between negative pairs in the feature space. Recent progress includes the introduction of the KAUR model, which enhances model efficiency through directly contrasting node embeddings from various graph views[22]. Additionally, the KGCL[4]and KGCCL[23]frameworks have been proposed for sequential recommendation. These leverage knowledge graph embeddings to better capture hierarchical structures in user behavior sequences, thereby enhancing the effectiveness of contrastive learning in recommendation systems. Furthermore, research on SGL-WA illustrates the advantages of minimal graph augmentations in optimizing recommendation systems[24].

In their pioneering work, Oord et al. introduced the Contrastive Predictive Coding (CPC) framework[25], which demonstrated the efficacy of feature learning via mutual information maximization. Robust theoretical groundwork is then provided for contrastive learning. Building upon the principles of CPC, this study designs three levels of contrastive learning as follows:

(1) At the user-user level, the analysis of behavioral pattern similarities among users, including through contrastive learning derived from users' historical activities, uncovers latent preference clusters. This approach effectively mitigates the issue of data scarcity at the user level.
(2) At the user-item level, the examination of users' varying preferences for different items enables a deeper understanding of their needs and preferences. Through using this approach, the precision of personalized recommendation systems can be significantly improved.
(3) At the item-item level, exploring and contrasting the interconnectedness among items, including recognizing similar products, not only addresses the issue of item-level sparsity but also augments the content diversity of recommendation systems.

By incorporating contrastive learning as a crucial element in the training process, this study further refines the quality of embeddings. This approach enhances the embeddings' discriminative prowess, which increases the accuracy and stability of the recommendation system, and strengthens its aptitude to handle sparse and varied data.

Method

Task definition

Before introducing the ML-KDGATMoco model, the following concepts and symbols are defined first.

User-Item Graph: User-item graph $\mathcal{G}_1$

is the interaction between users and items, and is represented by $\{(u, y_{ui}, i)|u \in \mathcal{U}, i \in \mathcal{I}\}$, where $\mathcal{U}$ is the user set, $\mathcal{I}$ is a collection of items and y_{ui} is the interaction data between users and items. Its definition is as follows:

$$y_{ui} = \begin{cases} 1, \text{ if interaction } (u,i) \text{ is observed;} \\ 0, \text{ otherwise.} \end{cases} \quad (1)$$

KG: The KG provides auxiliary information on items, which is recorded as $\mathcal{G}_2$. The (h, r, t) triplets represent a large number of entity-relationship-entity tuples in the KG, where $h \in \varepsilon, r \in R, t \in \varepsilon$ represent the head entity, relationship and tail entity in the triplet respectively.

CKG: The CKG refers to the combination of the user-item graph and knowledge graph, which encodes user behavior and item knowledge. The interaction behavior of each user is represented by $(u, Interact, i)$ triplets, where an additional connection between the user u and the item i is represented $y_{ui} = 1$. According to the item-entity alignment matrix, the user-item diagram can be seamlessly integrated with the KG as a unified diagram $\mathcal{G} = \{(h, r, t)|h, t \in \mathcal{E}'$, where $\mathcal{E}' \in \mathcal{E} \cup \mathcal{U}, \mathcal{R}' \in \mathcal{R} \cup \{Interact\}$.

Task definition: for a given collaborative KG $\mathcal{G}$ composed of $\mathcal{G}_1$ and $\mathcal{G}_2$, the recommendation task is defined as predicting whether a user u has a potential interest in an item i that has not been interacted with. This task aims to construct a prediction function $\widehat{y}_{ui} = F(u, i; \Theta)$, where $\widehat{y}_{ui}$ is the probability of interest of the user u in the item i and Θ represents the model parameter.

Model description

In this study, the ML-KDGATMoco model under the framework of self-supervised graph learning is designed. The primary task of this model lies in a Knowledge-aware Graph Attention Network (KDGAT). Self-supervised learning is used as an auxiliary task to enhance the model performance. The overall framework of the model is shown in Fig. 2. Among them, the self-supervised task is to generate two groups of variant subgraphs by randomly dropping the edges of a partial CKG, and then making contrastive learning at three levels: user-user interactions, item-item interactions and user-item interactions.

The contrastive learning model is mainly composed of the following five parts.

(1) Data enhancement layer: This layer is designed to enhance the generalization ability of the model by randomly dropping some edges of CKG, which can generate two groups of subgraphs with slight differences.

(2) Embedding layer: The model parameterizes each node (i.e. user or item) as an embedding vector while retaining the structural information of the CKG subgraph.

(3) Dynamic attention embedding dissemination layer: it is responsible for recursively propagating embedding from the neighbor of the node to update its representation. To capture different aspects of knowledge and weigh the importance of neighbors, the model adopts the dynamic attention mechanism of knowledge-aware.

(4) Nonlinear transformation layer: after the dynamic attention embedding dissemination layer is aggregated, the model further extracts features and enhances expression ability by passing the representations of users and items through a nonlinear transformation layer.

(5) Contrastive learning layer: The model performs the task of contrastive learning at three levels, and guides the model learning to distinguish positive and negative sample pairs by minimizing the contrast loss. This loss is combined with other losses to jointly optimize the model.

To facilitate the integration of contrastive learning in this graph attention framework, node embeddings produced by GATs are used to establish contrastive pairs. This strategic incorporation plays an important role in sharpening the model's focus on the discriminative attributes of nodes. The process involves generating positive pairs from adjacent nodes and negative pairs from non-adjacent nodes, which are dynamically updated through a momentum encoder. This tailored integration effectively boosts the model's ability to differentiate between similar and dissimilar nodes and significantly enhances the handling of sparse and dynamically evolving datasets.

Data enhancement layer

The data enhancement layer aims to provide training data for subsequent contrastive learning by generating two groups of CKG subgraphs enhanced by data. Different from CV and NLP fields, the enhancement of graph data should consider the complex connection between users and items. In this paper, the Edge Dropout[7] method, a graph data enhancement technology, is adopted. This method is realized by randomly removing the edges in the graph with a certain probability. The specific operation of this method is to define a mask vector of an edge. The edge is then selected randomly to be retained or deleted through this mask vector. The following is the corresponding formula:

$$s_1(\mathcal{G}) = (\mathcal{E}', \boldsymbol{A}_1 \odot \mathcal{R}'), s_2(\mathcal{G}) = (\mathcal{E}', \boldsymbol{A}_2 \odot \mathcal{R}') \tag{2}$$

where: $\boldsymbol{A}_1, \boldsymbol{A}_2 \in \{0,1\}^{|\mathcal{R}'|}$ are two mask vectors, which act on the edge set $\mathcal{R}'$ of CKG, while only some neighbors contribute to the node representation. By deleting some edges in the original CKG, two different subgraphs can be generated. Each subgraph captures the local structure in the original image. This data enhancement method

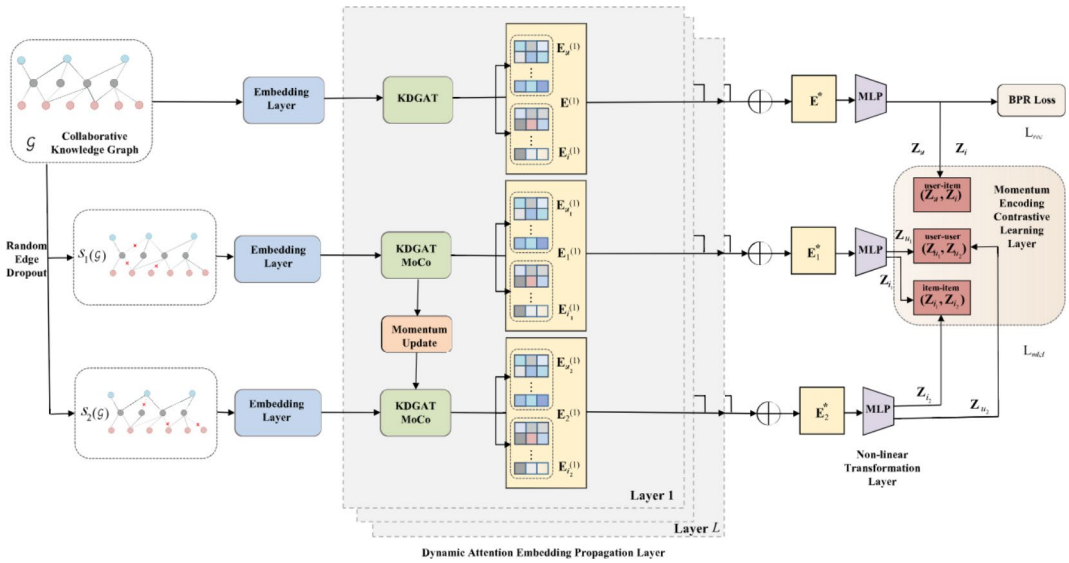

Fig. 2. Model Framework of ML-KDGATMoco.

not only improves the generalization ability of the model to unknown data but also enhances the robustness to potential noise and outliers.

Embedded layer

KGE is a technology that graphs entities and relationships within KG into vector representations. This technology aims to preserve semantic relationships and graph structures among entities in vector space. On CKG, this paper chooses the TransR[11] method, a translation model that graphs entities and relationships to different vector spaces. TransR optimizes the vector representation of entities and relationships by treating the tail entity as a "translation" of the head entity and the relationship vector. This process can be understood as the transformation of entities on the "bridge" of relations. Specifically, with a given triple (h, r, t), the TransR method first graphs the vector e_h of the head entity h and the vector e_t of the tail entity t to the vector space of the relationship r. The model then learns the vector representation of the entity and the relationship by minimizing the "translation" distance between the head entity and the tail entity in the vector space of the relationship.

Where: e_h^r and e_t^r represent the projection of e_h and e_t in the contact r space. The scoring function is:

$$g(h, r, t) = ||\boldsymbol{W}_r e_h + e_r - \boldsymbol{W}_r e_t||_2^2 \tag{3}$$

where:$e_h, e_t \in \mathbb{R}^d$ and $e_r \in \mathbb{R}^k$ represent the embedment of h, t and r respectively. As a relational transformation matrix of contact r, $\boldsymbol{W}_r \in \mathbb{R}^{k \times d}$ linearly transforms entities from the original d dimensional entity space to the k dimensional relationship vector space. A lower value of $g(h, r, t)$ indicates that the triple (h, r, t) is more likely to exist in the KG.

When training the TransR model, both positive and negative examples should be considered to encourage the distinction between them. The loss function is defined as:

$$\mathcal{L}_{KG} = \sum\nolimits_{(h,r,t,t') \in T} - ln\sigma\left(g\left(h, r, t'\right) - g\left(h, r, t\right)\right) \tag{4}$$

where:$T = \{(h, r, t, t\prime)|(h, r, t) \in \mathcal{G}, (h, r, t\prime) \notin \mathcal{G}\}$. (h, r, t') represents a negative example, which is a negative sample set obtained by negative sampling. $\sigma(\cdot)$ is the *sigmoid*function. Compared with other KGE methods, including TransE[10], the advantage of TransR is that it can better handle the complex relationship between entities and thus is more suitable for the embedding layer of this model. By mapping entities and relationships to different vector spaces, TransR improves the representation ability of the model to the KG structure and enhances the regularization effect of the direct relationship between entities in vector representation.

Dynamic attention embedding dissemination layer

Dynamic attention embedding dissemination layer is an architecture based on the graph convolution network[26]. It captures the high-order connectivity between entities in the KG by recursively propagating the embedding information of entities. Each layer consists of three core modules: information dissemination, knowledge-aware dynamic attention calculation and information aggregation.

Information dissemination module: In this module, this paper assumes that an entity can appear in multiple triples and disseminate information through these triples. For an entity h, all the triple sets $\mathcal{N}_h = \{(h, r, t)|(h, r, t) \in \mathcal{G}\}$ that it participates in with h as the head entity are defined as the neighbors of h, called the neighbor vector[27] of h. Each neighbor is connected by triplets and transmits information through the following linear combinations:

$$e_{\mathcal{N}_h} = \sum\nolimits_{(h,r,t) \in \mathcal{N}_h} \pi\left(h, r, t\right) e_t \tag{5}$$

where: $\pi(h, r, t)$ is the attenuation factor that controls the information dissemination on each edge (h, r, t), which represents how much information propagates from t to h under the relation r condition.

Knowledge-aware dynamic attention calculation module: In the traditional GAT, the attention weight of each node only depends on its neighbor nodes. This paper adopts the dynamic attention mechanism proposed in reference[6], which allows the weight ranking of neighboring nodes to be updated accordingly with the update of query nodes. At the same time, to enhance the model's stability and its adaptability to complex data structures, layer normalization and residual networks are introduced into the dynamic attention mechanism. Before the calculation of the attention coefficients, each node in the graph, including the head node e_h, tail node e_t, and the relational vector e_r, uniformly performs residual connections and layer normalization:

$$e'_x = LayerNorm(e_x + Transform(e_x)) \tag{6}$$

where: e'_x denotes the original embedding vector of the node x, which can be either a head node e_h, a tail node e_t , or a relational vector e_r. The function $LayerNorm(\cdot)$ and $Transform(\cdot)$ corresponds to layer normalization and residual connection operations, respectively. Layer normalization aims to standardize the features across each layer to stabilize learning, while residual connection operations allow the flow of information and gradients through the network, which mitigates the vanishing gradient problem. These operations are integral to enhancing the model's capability to process complex data structures and guarantee robust learning dynamics.

By adopting optimized node and contact vector representations, the attenuation factor formula for calculating the dynamic attention weight is as follows:

$$\pi(h,r,t) = a^T LeakyReLU(\boldsymbol{W} \cdot [e'_h||e'_t||e'_r]) \tag{7}$$

where: the nonlinear activation function is LeakyReLU[28], and a and $\boldsymbol{W}$ are learnable parameters. || indicates the splicing operation of vectors. Then, to ensure that these scores are comparable to the distribution of neighbors, all attenuation factors are normalized by the softmax function to obtain the final dynamic attention weight:

$$\pi(h,r,t) = \frac{exp(\pi(h,r,t))}{\sum_{(h,r',t')\in\mathcal{N}_h} exp(\pi(h,r',t'))} \tag{8}$$

Information aggregation module: In the last module, the entity representation e_h and neighbor vector representation $e_{\mathcal{N}_h}$are aggregated by using the Bi-Interaction aggregator[3]. That allows the embedding of the head entity to interact with the embedding of its neighbors and updates the representation of the head entity. The specific definition of $f(\cdot)$ is as follows:

$$f_{Bi-Interaction} = LeakyReLU\left(\boldsymbol{W}_1(e_h + e_{\mathcal{N}_h})\right) + LeakyReLU\left(\boldsymbol{W}_2(e_h \odot e_{\mathcal{N}_h})\right) \tag{9}$$

where: $\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{d\prime\times d}$ is a learnable weight matrix, $d\prime$ is the conversion size, and $\odot$ represents the product at the element level. In this way, each layer of dynamic attention embedding communication layer can adjust the way of information dissemination according to the neighbor situation of the current entity. Accordingly, the network can flexibly adapt to the structural changes in the KG.

By stacking multiple such dissemination layers, information on neighbor nodes with higher hops can be collected. After l layers of iteration, the final representation of the header entity h is updated to:

$$e_h^{(l)} = f\left(e_h^{l-1}, e_{\mathcal{N}_h}^{(l-1)}\right) \tag{10}$$

where: $e_{\mathcal{N}_h}^{(l-1)}$ is the neighbor vector representation of the l-th layer head entity, which involves the information disseminated from multi-hop neighbor nodes:

$$e_{\mathcal{N}_h}^{(l-1)} = \sum_{(h,r,t)\in\mathcal{N}_h} \pi(h,r,t) e_t^{(l-1)} \tag{11}$$

where: $e_t^{(l-1)}$ is the representation of the tail entity t generated by the previous information dissemination step.

Nonlinear transformation layer

After performing the L aggregation layers, multiple hierarchical representations of the user u and item i in the original CKG are obtained, namely $\{e_u^{(1)}, \cdots, e_u^{(L)}\}$ and $\{e_i^{(1)}, \cdots, e_i^{(L)}\}$. The multi-level representation of the user nodes u_1, u_2, i_1 and i_2 of $s_1(\mathcal{G})$ and $s_2(\mathcal{G})$enhanced by two groups of ED data are obtained. To enhance the expressive ability of these representations, the experiment follows the method in reference[3] and adopts the layer aggregation mechanism, for fusing the user and item representations of each layer into a comprehensive representation through a splicing operation:

$$\begin{aligned}
e_u^* &= e_u^{(1)} \parallel \cdots \parallel e_u^{(L)}, e_i^* = e_i^{(1)} \parallel \cdots \parallel e_i^{(L)} \\
e_{u_1}^* &= e_{u_1}^{(1)} \parallel \cdots \parallel e_{u_1}^{(L)}, e_{i_1}^* = e_{i_1}^{(1)} \parallel \cdots \parallel e_{i_1}^{(L)} \\
e_{u_2}^* &= e_{u_2}^{(1)} \parallel \cdots \parallel e_{u_1}^{(L)}, e_{i_2}^* = e_{i_2}^{(1)} \parallel \cdots \parallel e_{i_2}^{(L)}
\end{aligned} \tag{12}$$

Given that contrastive learning aims at maximizing the consistency of positive sample pairs, information beneficial to recommendation tasks can be sacrificed in the optimization process. To alleviate this problem, as inspired by SimCLR[20], a learnable nonlinear transformation layer $g(\cdot)$ is introduced before the comprehensive representation is sent to the contrastive learning layer. This transformation layer is realized by a multi-layer perceptron (MLP) composed of a hidden layer with a ReLU activation function:

$$z_{input} = g(e_{input}^*) = \boldsymbol{W}^{(2)}\sigma(\boldsymbol{W}^{(1)} e_{input}^*) \tag{13}$$

where: $\sigma(\cdot)$ represents the ReLU nonlinear activation function. e_{input}^* and $input \in \{u, u_1, u_2, i, i_1, i_2\}$ are the representations of the converted user and item. After $g(\cdot)$ transformation, z_u, z_{u_1}, z_{u_2}, z_u, z_i, z_{i_1} and z_{i_2} are obtained, which are used for subsequent contrastive learning respectively.

Through introducing a nonlinear transformation layer, the information of the aggregation layer is retained. In addition, the expression ability of the model before applying comparative loss is enhanced, which helps improve the performance of the final recommendation system.

Contrastive learning layer

Contrastive learning relies on the number of negative samples to generate high-quality representations. For example, SimCLR[20]proposed an end-to-end model, which was trained in large quantities (including 4096). However, a too-large batch greatly consumes memory and video memory because each additional sample leads to a significant increase in model parameters. Hence, it may not be conducive to optimizing the training process. To solve this problem, Memory Bank[29]proposed an independent dictionary to store the representations of all samples as negative samples and update them regularly. Nevertheless, frequent updates of representations in Memory Bank may bring vector inconsistency and affect the uniformity of the model. For this reason, Moco[21] adopted a contrastive learning strategy which includes a dynamic queue and momentum encoder. Despite the consistency of updating parameters of the momentum encoder and query encoder, the updating speed is slow. The queue adopts the "first in first out" mode, which ensures the continuity and consistency of the dictionary. Based on this, the Moco is introduced to construct high-quality negative samples and enhance the model's feature learning ability.

In this study, a novel contrastive learning framework designed to operate across three distinct levels, including user-user, item-item, and user-interactive item, is introduced. Each level targets specific aspects of recommendation systems to refine and enhance predictive accuracy and personalization capabilities. Taking the user-to-user contrast as an example, the KDGATMoCo algorithm is developed, which effectively captures the differences between user characteristics through the Momentum Encoder. The implementation details of KDGATMoCo are shown in Algorithm 1.

```
# Initialize
# kdgat_q, kdgat_k: encoders of Q and K
# queue: query dictionary, including queues of K keys (CxK)
# C: embed the dimension
#K: quantity of negative samples
#x_q: minibatch including N samples, which are randomly sampled from Z_u1
#x_k: minibatch including N samples, which are randomly sampled from Z_u2
kdgat_k.params =  kdgat_q.params # initialize
#forward dissemination
q = kdgat_q.forward(x_q) #query: NxC
k = kdgat_k.forward(x_k) # key: NxC
k = k.detach() # gradient of detachment key
# calculate contrastive loss
l_pos = bmm(q.view(N,1,C), k.view(N,C,1))# positive sample logarithm: Nx1
l_neg = mm(q.view(N,C), queue.view(C,K))# nagetive sample logarithm: NxK
logits = cat([l_pos, l_neg], dim=1)#logarithm: Nx(1+K)
# calculate loss and perform backward dissemination
labels = zeros(N) # positive sample label is 0
loss = CrossEntropyLoss(logits/t, labels) #contrastive loss
loss.backward() # backward dissemination
update(kdgat_q.params) # SGD update Q encoder
# momentum update and queue update
kdgat_k.params=m*kdgat_k.params+(1-m)*kdgat_q.params # momentum update K encoder
enqueue(queue, k) # queue minibatch enters now
dequeue(queue) # dequeue the earliest minibatch
```

Algorithm 1. KDGATMoCo.

At first, KDGATMoCo initializes two encoders (including kdgat_q and kdgat_k) respectively to generate the representation of the query and key. These encoders are updated synchronously through the MoCo. Specifically,

the momentum coefficient is set to 0.999 by default to ensure the consistency and stability of key representation. The formula of the momentum renewal mechanism is expressed as:

$$\theta_k \leftarrow m\theta_k + (1-m)\,\theta_q \tag{14}$$

where: θ_q is the parameter of the query encoder, θ_k is the parameter of the keys encoder, and $m \in [0,1)$ is the momentum update coefficient.

In the case of user-user contrastive learning, the representations of user nodes are utilized as queries, and a group of user representations are taken as keys in the dictionary. The model is optimized by the InfoNCE[30] loss function, a pivotal choice for its ability to modulate the similarities between sample representations via a finely tuned temperature hyperparameter T. InfoNCE loss is defined as:

$$\mathcal{L}_q = -log\frac{exp(q \cdot k_+/T)}{\sum_{i=0}^{K} exp(q.k_i/T)} \tag{15}$$

where: K is the negative sample number, k_+ is the key that matches the query , and T is the temperature hyperparameter.

The total loss of user-user contrastive learning is calculated as:

$$\mathcal{L}_{cl}^{u-u} = \sum\nolimits_{j=0}^{K} \mathcal{L}_{q_j} \tag{16}$$

In the same way, the loss of item-item contrastive learning $\mathcal{L}_{cl}^{i-i}$ and the loss of user-interactive item contrastive learning $\mathcal{L}_{cl}^{u-i}$ can be calculated. Finally, the total loss of the model is formed by adding up the contrastive learning losses of these three levels to guide the training of the model:

$$\mathcal{L}_{mlcl} = \mathcal{L}_{cl}^{u-u} + \mathcal{L}_{cl}^{i-i} + \mathcal{L}_{cl}^{u-i} \tag{17}$$

Therefore, KDGATMoCo can effectively employ the user and item information in the KG, and improve the accuracy and efficiency of the recommendation system.

Joint training loss function

To optimize the ML-KDGATMoCo model, a joint loss function is adopted. The function combines the main loss of the recommendation model, the KG training loss of the embedded layer, and three groups of contrastive learning losses. The loss function of joint optimization can be expressed as:

$$\mathcal{L}_{joint} = \lambda_1 \mathcal{L}_{rec} + \lambda_2 \mathcal{L}_{KG} + \lambda_3 \mathcal{L}_{mlcl} + \lambda_4 \parallel \Theta \parallel_2^2 \tag{18}$$

where: Θ represents the model parameter set, L_2 is a regularization term, and λ_1,λ_2,λ_3,λ_4 are the loss function coefficients that balance the contributions of each loss component to optimize overall model performance.

$\mathcal{L}_{rec}$is the main loss function. The widely used BPR loss[31] is selected as the main loss, with the following calculation formula:

$$\mathcal{L}_{rec} = \sum\nolimits_{(u,i,j)\in\mathcal{O}} - ln\sigma\left(\widehat{y}(u,i) - \widehat{y}(u,j)\right) \tag{19}$$

where: $\mathcal{O} = \{(u,i,j)\,|(u,i) \in \mathcal{R}^+, (u,j) \in \mathcal{R}^-\}$ represents the training set, and $\sigma(\cdot)$ is the *sigmoid* function. $\widehat{y}(u,i)$ calculates the similarity between the user and the item after all the dissemination layers are aggregated. In other words, their matching score is as follows:

$$\widehat{y}\,(u,i) = e_u^{*T} e_i^* \tag{20}$$

Through designing this joint loss function, the model can not only effectively learn the feature representation of users and items, but also meaningfully employ the structured information in the KG. Accordingly, more accurate and personalized predictions can be realized in the recommendation system.

Experiment and analysis

In this paper, the performance of the ML-KDGATMoco model is verified on the RecBole[32] platform. The experimental environment is a single GPU(RTX 2080 Ti), with a CPU of Intel Core i7-8700K@4.7 GHz and memory of 128G.

Dataset

Through employing the two datasets, ML-1m_KG and Amazon-books-KG, based on KB4Rec[33], this paper combines the dataset of MovieLens and Amazon-books with the Knowledge Graph Freebase (KGF). These datasets are preprocessed by 15-core filtering and 5-core filtering to improve data quality. Table 1 summarizes the statistical information of the preprocessed dataset.

Dataset	ML-1m-KG	Amazon-books-KG
#Users	5986	20,510
#Item	2975	21,739
#Interactions	832,045	886,543
#Entity	79,363	129,798
#Relation	51	22
#Triple	385,923	361,741

Table 1. Statistics of the datasets.

Contrast model

In the experiment, several representative models in RecBole are selected for contrast as follows:

(1) **SGL**[7]: This is an innovative contrastive learning recommendation model, which generates enhanced views through random walk sampling and random edge/node loss technology to enhance the learning of user preferences.

(2) **KGCN**[2]: As a model based on a GNN, KGCN uses a KG as auxiliary information to improve the accuracy and interpretability of the recommendation system.

(3) **SimGCL**[24]: By directly adding uniform noise to embeddings, which creates contrastive views, SimGCL smoothly adjusts representation uniformity, thereby demonstrating significant advantages in recommendation accuracy and training efficiency over augmentation-based methods.

(4) **KGAT**[3]: KGAT uses the GNN to model the complex relationships in the KG, assists the recommendation process, and emphasizes the in-depth utilization of the KG structure.

(5) **XSimGCL**[34]: This model uses a simple, effective noise-based method to create contrastive samples, enhancing accuracy while reducing computational costs.

Experimental evaluation indicators and experimental settings

In the experiment, the dataset is preprocessed. 80% of each user's interaction records are randomly selected as the training set, and the remaining 20% as the test set. In addition, 10% of the interaction records are randomly selected from the training set as the verification set for tuning the hyperparameter. In this process, all items that the user has not interacted with are regarded as negative samples.

To comprehensively evaluate the effectiveness of the Top-K recommendation, various evaluation indicators are adopted, including Recall@K, MRR@K, NDCG@K, Hit@K and Precision @K. Specifically, the value of k is set to 10. Choosing $K=10$ is based on the characteristics of the selected dataset-music and book recommendation. In this kind of recommendation scenario, users tend to browse more recommended content to find items that they may be interested in. Therefore, improving the performance of Hit@10 is more practical than improving Hit@1. All evaluation indicators are calculated based on the average of all users in the test set.

To ensure the fairness and comparability of the experiment, the parameter settings of all baseline models follow the recommended configuration officially recommended by RecBole. Specifically, the user and item embedding dimensions of the two datasets are set to 64, with the training batch size of 4096, and the training epoch set to 500. For the ML-KDGATMoco framework, the parameter setting of the main supervision module is consistent with the KGAT model.

In the self-supervised contrastive learning module, the initial value of random Edge Dropout is set to 0.2, the parameter of the loss function is set to 0.01, the parameter of the loss function is set to 0.01, the parameter of temperature is set to 1, and the optimizer is Adam. In addition, an early stop strategy is adopted. This means that if Recall@10 on the verification set is not improved in 50 consecutive epochs, then the training will be terminated early.

Analysis of experimental results

The experimental results are shown in Table 2. Compared with the other four baseline models, the ML-KDGATMoco model indicates significant performance improvement in each evaluation indicator. In particular, on the sparse Amazon-books-KG dataset, compared with the baseline model, the proposed model is improved by 15.6%, 15.8%, 16.6%, 8.6% and 9.3% on Recall@10, NDCG@10, Hit@10 and Precision @10, respectively. This result highlights the powerful ability of ML-KDGATMoco to handle sparse datasets.

On the ML-1m_KG dataset, the KGAT model performs best in most performance indicators. However, on the more challenging and sparser Amazon-book-KG dataset, XSimGCL models show obvious advantages. Through the self-supervised graph learning method, the XSimGCL model effectively uses graph structure information to learn the representation of nodes and edges. Therefore, it performs well in handling sparse problems within large-scale KGs.

In contrast, the KGAT model may face the challenge of learning effective entity and relationship information in extremely sparse datasets. Although its attention mechanism contributes to identifying the importance of entities and relationships, it may not be efficient enough in sparse datasets with a small number of samples. Thus, it has difficulties in allocating attention.

Compared with these models, the outstanding performance of ML-KDGATMoco on Amazon-book-KG datasets is mainly due to its multi-level contrastive learning and optimization strategy for sparse data. This

Dataset	Indicator	SGL	KGCN	SimGCL	KGAT	XSimGCL	ML-KDGATMoco
ML-1m-KG	R@10	0.1560	0.1561	0.1608	0.1657	0.1643	**0.1797 (+8.5%)**
	M@10	0.3929	0.3927	0.3935	0.4085	0.4065	**0.4259 (+4.3%)**
	N@10	0.2191	0.2191	0.2236	0.2326	0.2345	**0.2457 (+4.8%)**
	H@10	0.6911	0.6914	0.7016	0.7089	0.7068	**0.7368 (+3.9%)**
	P@10	0.1666	0.1667	0.1702	0.1755	0.1717	**0.1842 (+5.0%)**
Amazon-books-KG	R@10	0.0535	0.0548	0.0557	0.0523	0.0585	**0.0676 (+15.6%)**
	M@10	0.0502	0.0489	0.0509	0.0500	0.0531	**0.0615 (+15.8%)**
	N@10	0.0358	0.0357	0.0373	0.0359	0.0391	**0.0456 (+16.6%)**
	H@10	0.1395	0.1386	0.1426	0.1326	0.1482	**0.1609 (+8.6%)**
	P@10	0.0160	0.0159	0.0166	0.0151	0.0172	**0.0188 (+9.3%)**

Table 2. Performance comparison with baselines on two datasets. Significant values are in underline and bold.

Dataset	Indicator	MLCL-d	MLCL-m	MLCL-p
ML-1m-KG	R@10	0.1742	0.1719	0.1744
	M@10	0.4232	0.4178	0.4194
	N@10	0.2431	0.2387	0.2396
	H@10	0.7280	0.7180	0.7272
	P@10	0.1820	0.1784	0.1788
Amazon-books-KG	R@10	0.0640	0.0647	0.0667
	M@10	0.0590	0.0603	0.0602
	N@10	0.0433	0.0441	0.0452
	H@10	0.1548	0.1556	0.1559
	P@10	0.0177	0.0180	0.0183

Table 3. Results of the ablation study.

result shows that the combination of contrastive learning and KG can effectively improve the performance of the recommendation system in handling sparse data.

Ablation experiment

To verify that the self-supervised learning task of contrastive learning and its sub-models in ML-KDGATMoco can effectively improve the whole recommendation performance, some modules are deleted respectively. Three variants, including MLCL-d, MLCL-m and MLCL-p, are obtained. Specifically, MLCL-d is the contrastive learning using the static attention network in KGAT. MLCL-m is contrastive learning using the common negative sample strategy based on the batch. MLCL-p is the contrastive learning without the nonlinear transformation layer.

The three variants are put into two datasets for test experiments again. The results are shown in Table 3.

It can be concluded:

1) All the performance indicators of the Contrastive Learning Variant (MLCL-d), which replaces the dynamic attention network with the static attention network in 2 datasets, have declined to different degrees. In particular, on the Amazon-books-KG dataset, each performance indicator decreases by 3.79–5.85 points. The impacts on indicators Recall@10, NDCG@10 and Precision@10.

are over 5 points. The result shows that the dynamic attention mechanism can better model the KG with sparse data than the static attention mechanism, which plays an important role in improving model performance.

2) Removing the Moco module or the nonlinear transformation layer can reduce the performance of the two datasets. Relatively speaking, the Moco module exerts a great impact on the performance. The effects reach 1.90–3.41 points on each performance indicator of the ML-1m_KG dataset and 1.95–4.26 points on each performance indicator of the Amazon-books-KG dataset. However, the nonlinear transformation layer affects about 1.30–2.93 points on each performance indicator of the ML-1m_KG dataset, and about 0.88–3.11 points on each performance indicator of the Amazon-books-KG dataset. It is verified that introducing a momentum encoder to expand negative samples and a nonlinear transformation layer to retain more feature information can play a positive role in contrastive learning.

Data sparsity experiment

To test whether ML-KDGATMoco is robust to the data sparsity problem common in knowledge-aware recommendation systems, experiments are conducted on user groups with different sparsity levels. The test set is divided into four groups according to the number of user interactions. Different groups are kept with the same total amount of interactions as much as possible. Taking the ML-1m_KG dataset as an example, it is divided into groups with less than 148 interactions, less than 302 interactions, less than 543 interactions and less than 2315

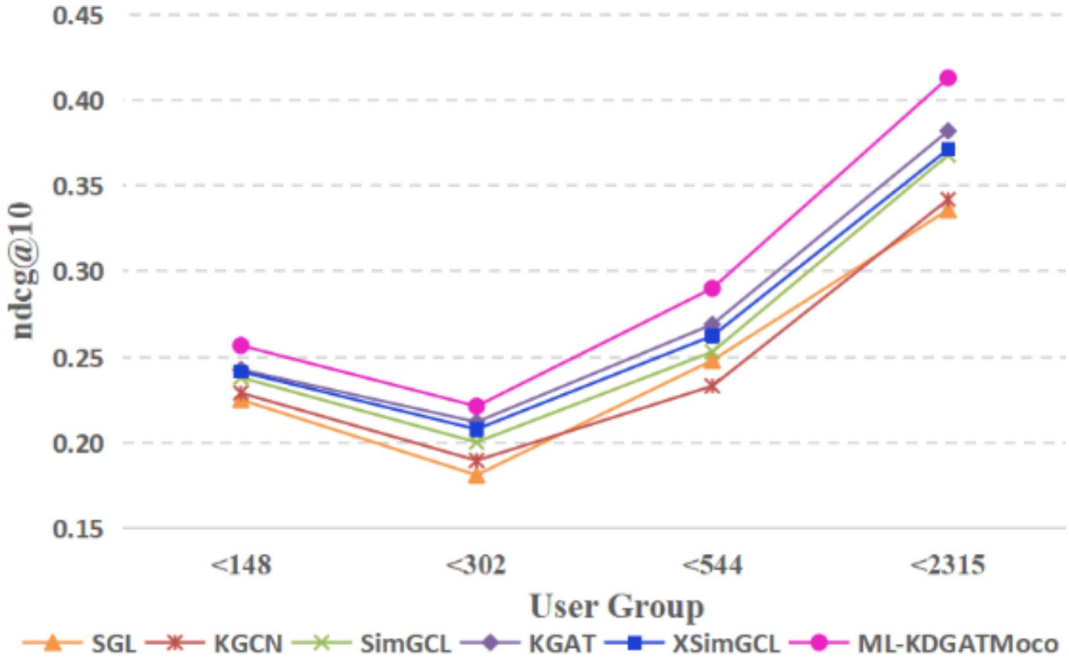

Fig. 3. Performance comparison of the data sparsity on ML-1m_KG.

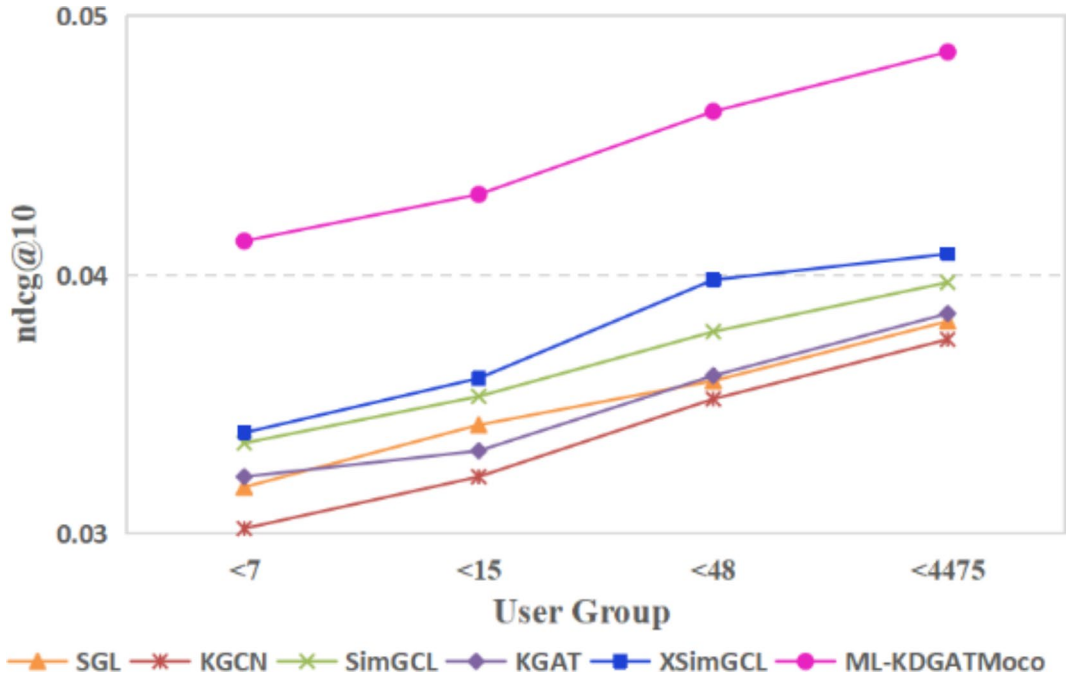

Fig. 4. Performance comparison of the data sparsity on Amazon-books-KG.

interactions, separately. Figures 3 and 4 show the experimental results of NDCG@10 for different user groups on two datasets. As can be seen, ML-KDGATMoco can still learn the node representation well in the case of extremely sparse data. The performance of maintaining NDCG is better than the baseline model, and the more sparse the dataset, the more obvious this advantage is.

Parameter discussion

The ML-KDGATMoco model contains two key hyperparameters: the dropout rate ρ of edge dropout and the temperature hyperparameter T of the InfoNCE loss function. The dropout rate of edge dropping controls the degree of edge dropout of the data enhancement module in contrastive learning. The larger the value, the greater

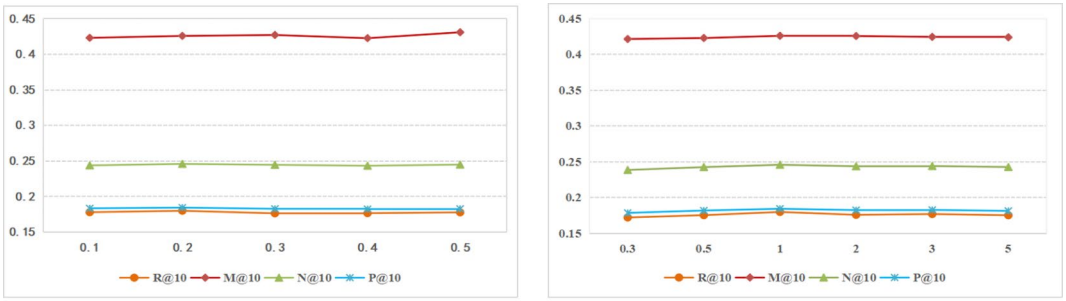

Fig. 5. Performance change of hyperparameters on two datasets. (**a**) Performance change of ρ parameter. (**b**) Performance change of T parameter.

the difference between the newly constructed graph and the original graph, and the smaller the difference. The temperature hyperparameter affects the sensitivity of the InfoNCE loss function. By changing the values of these two parameters and observing the performance changes of the model on ML-1m-KG and Amazon-books-KG datasets, the influences on the model performance can be evaluated.

The experiment results (Fig. 5) show that though the fine-tuning of these parameters can improve the performance, the overall performance of the model is affected by other factors. These factors can be model architecture, quality and complexity of training data. The performance indicator shows a trend of increasing first and then decreasing with the change of parameters. This indicates that parameter adjustment should be refined to achieve optimal performance.

Conclusion

By combining a KG with a recommendation system, this study explores the application possibilities of contrastive learning in knowledge-aware recommendation. Firstly, the KG and user-item interaction is regarded as a whole graph structure. Three levels of contrastive learning tasks are designed to train jointly with recommendation tasks. In addition, by introducing a momentum encoder and dynamic attention mechanism, the performance of the model is effectively improved when processing sparse data. The experimental results on two real datasets confirm the superior performance of the ML-KDGATMoco model in the field of knowledge-aware recommendation, especially its powerful ability to handle sparse datasets. In the future, the research will further focus on the new application of contrastive learning in the recommendation system based on KGs.

Data availability

Data is provided within the manuscript or supplementary information files.

Received: 28 March 2024; Accepted: 26 September 2024
Published online: 04 October 2024

References

1. Wu, S. *et al.* Graph neural networks in recommender systems: a survey[J]. *ACM Comput. Surv.***55**(5), 1–37 (2022).
2. Wang Hongwei, Zhao Miao, Xie Xing, *et al.* Knowledge graph convolutional networks for recommender systems [C]// The world wide web conference. 2019: 3307–3313.
3. Wang Xiang, He Xiangnan, Cao Yixin, *et al.* KGAT: Knowledge graph attention network for recommendation [C]// Proc of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. 2019: 950–958.
4. Yang Yuhao, Huang Chao, Xia Lianghao, *et al.* Knowledge graph contrastive learning for recommendation [C]// Proc of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2022: 1434–1443.
5. Zou Ding, Wei Wei, Wang Ziyang, *et al.* Improving knowledge-aware recommendation with multi-level interactive contrastive learning [C]// Proc of the 31st ACM International Conference on Information & Knowledge Management. 2022: 2817–2826.
6. Wu Jiancan, Wang Xiang, Feng Fuli, *et al.* Self-supervised Graph Learning for Recommendation [C]// SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2021, 726–735.
7. Zhu Yanqiao, Xu Yichen, Yu Feng, *et al.* Graph Contrastive Learning with Adaptive Augmentation [C]// WWW'21: The Web Conference 2021. 2021.
8. Ai Qingyao , Vahid Azizi, Chen Xu , *et al.* Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation. [J]. Algorithms, 2018, **11** (9): 137.
9. Bordes A, Usunier N, Garcia-Duran A, *et al.* Translating embeddings for modeling multi-relational data[J]. Advances in neural information processing systems, 2013, 26.
10. Zhang Fuzheng, Yuan N J, Lian Defu, *et al.* Collaborative knowledge base embedding for recommender systems [C]// Proc of the 22th ACM SIGKDD international conference on knowledge discovery and data mining. 2016: 353–362.
11. Lin Yankai, Liu Zhiyuan, Sun Maosong, *et al.* Learning entity and relation embeddings for knowledge graph completion [C]// Proc of the AAAI conference on artificial intelligence. 2015, **29** (1):2181–2187.
12. Geng Shijie, Fu Zuohui, Tan Juntao, *et al.* Path language modeling over knowledge graphs for explainable recommendation [C]// Proc of the ACM Web Conference. 2022: 946–955.
13. Jaiswal, A. *et al.* A survey on contrastive self-supervised learning[J]. *Technologies***9**(1), 2 (2020).
14. Wang Xiting, Liu Kunpeng, Wang Dongjie, et al. Multi-level recommendation reasoning over knowledge graphs with reinforcement learning [C]//Proc of the ACM Web Conference. 2022: 2098–2108.

15. Guo Shipeng, Liu Kunpeng, Wang Pengfei, et al. RDKG: A Reinforcement Learning Framework for Disease Diagnosis on Knowledge Graph [C]//Proc of the IEEE Conference on Data Mining (ICDM). 2023: 1049–1054.
16. Zhang S, Chen L, Wang C, *et al.* Temporal Graph Contrastive Learning for Sequential Recommendation [C]// Proc of the AAAI Conference on Artificial Intelligence. 2024, **38** (8): 9359–9367.
17. Wei Z, Wang K, Li F, *et al.* M3KGR: A Momentum Contrastive Multi-Modal Knowledge Graph Learning Framework for Recommendation[J]. Information Sciences, 2024: 120812.
18. Herlocker J L, Konstan J A, Borchers A, et al. An algorithmic framework for performing collaborative filtering [C]// Proc of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. 1999: 230–237.
19. Koren, Y., Bell, R. & Volinsky, C. Matrix factorization techniques for recommender systems[J]. *Computer***42**(8), 30–37 (2009).
20. Xia L, Huang C, Huang C, et al. Automated SelfSupervised Learning for Recommendation[C]// Proc of the ACM Web Conference. 2023: 992–1002.
21. He Kaiming, Fan Haoqi, Wu Yuxin, *et al.* Momentum contrast for unsupervised visual representation learning [C]// Proc of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 9729–9738.
22. Ma, Y. *et al.* Enhancing recommendations with contrastive learning from collaborative knowledge graph[J]. *Neurocomputing***523**, 103–115 (2023).
23. Meng Z, Ounis I, Macdonald C, *et al.* Knowledge Graph Cross-View Contrastive Learning for Recommendation [C]// European Conference on Information Retrieval. Cham: Springer Nature Switzerland, 2024: 3–18.
24. Yu Junliang, Yin Hongzhi, Xia Xin, et al. Are graph augmentations necessary? simple graph contrastive learning for recommendation [C]// Proc of the 45th international ACM SIGIR conference on research and development in information retrieval. 2022: 1294–1303.
25. Oord A, Li Y, Vinyals O. Representation learning with contrastive predictive coding[J]. arXiv preprint arXiv:1807.03748, 2018.
26. Kipf T N, Welling M. Semi-Supervised Classification with Graph Convolutional Networks [C]// International Conference on Learning Representations. 2017.
27. Qiu Jiezhong, Tang Jian , Ma Hao, *et al.* Deepinf: Social influence prediction with deep learning [C]// Proc of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. 2018: 2110–2119.
28. Maas, A. L., Hannun, A. Y. & Ng, A. Y. Rectifier nonlinearities improve neural network acoustic models. *In ICML***30**, 3 (2013).
29. Wu Zhirong, Xiong Yuanjun, Yu S X, *et al.* Unsupervised feature learning via non-parametric instance discrimination [C]// Proc of the IEEE conference on computer vision and pattern recognition. 2018: 3733–3742.
30. Aaron van den Oord, Li Yazhe , Oriol Vinyals. Representation learning with contrastive predictive coding[J]. arXiv:1807.03748, 2018.
31. Rendle S, Freudenthaler C, Gantner Z, *et al.* BPR: Bayesian personalized ranking from implicit feedback [C]// Proc of the 25th Conference on Uncertainty in Artificial Intelligence. 2009: 452–461.
32. Zhao Wayne Xin, Mu Shanlei, Hou Yupeng, *et al.* Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms [C]// Proc of the 30th ACM International Conference on Information & Knowledge Management. 2021: 4653–4664.
33. Zhao Wayne Xin, He Gaole, Dou Hongjian, *et al.* Kb4rec: A dataset for linking knowledge bases with recommender systems[J]. Data Intelligence, 2019, **1**(2): 121–136.
34. Yu, J. *et al.* XSimGCL: Towards extremely simple graph contrastive learning for recommendation[J]. *IEEE Transactions on Knowledge and Data Engineering***36**(2), 913–926 (2023).

Author contributions

Rong Zhang and Yang Li wrote the main manuscript text and Yuan Liu prepared figures 1-2. All authors reviewed the manuscript.

Funding

Supported by National Natural Science Foundation of China (61972182); Project support for the construction of high-level professional groups in higher vocational education in Jiangsu Province (S.J.Z.H. [2021] No.1); Excellent scientific and technological innovation team of colleges and universities in Jiangsu Province (S.J.K. [2023] No.3); Engineering Technology Research and Development Center of Jiangsu Higher Vocational Colleges (S.J.K.H. [2023] No.11).

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-024-74516-z.

Correspondence and requests for materials should be addressed to Z.R.