Digital Dissemination of Information Based on Knowledge Graph Recommendation Algorithm

Lu Wang[1, 2*], Yunzhen Zhang[3]
[1]Department of Investigation, Henan Police College, Zhengzhou 450046, China
[2]College of Computer Science and Information Engineering, Anyang Institute of Technology, Anyang 455000, China
[3]School of Energy and Intelligence Engineering, Henan University of Animal Husbandry and Economy, Zhengzhou 450046, China
E-mail: Wanglu278650710@163.com
[*]Corresponding author

Keywords: knowledge graph, graph neural network, recommendation algorithm, ripple transmission network, feature extraction

Received: July 4, 2024

The rapid advancement of technology has made recommendation systems a research focus in modern information processing. To enhance the accuracy of recommendation systems and effectively capture and recommend user interests in complex data environments, this study starts with the combination of knowledge graphs and graph neural networks. Firstly, the water wave network algorithm is introduced to diffuse user interest information and fully utilize the global information of the knowledge graph. Secondly, the processed knowledge graph is input into the graph neural network for deep feature learning, and finally, a recommendation algorithm combining the ripple network and graph neural network is proposed. The test results showed that when the final recommendation results of the model were 5, 10, 15, 20, 25, and 30, the matching degrees of the recommendation results were 99.2%, 98.3%, 97.4%, 96.5%, 95.8%, and 95.2%, respectively. On the 100% dataset, the improved model had a 7.52% higher hit rate than the original knowledge graph convolutional network, and the mean reciprocal ranking value increased by 14.29. The normalized discounted cumulative gain on different datasets has increased by an average of 7.0%. Experiments have shown that by combining knowledge graphs and graph neural networks, the dynamic adaptability and recommendation quality of recommendation systems have been effectively improved, providing an efficient and accurate solution for information recommendation.

Povzetek: Raziskava predstavlja nov priporočilni algoritem, ki združuje grafe znanja in nevronske mreže na grafih za izboljšanje kvalitete in prilagodljivosti sistemov za priporočanje.

1 Introduction

In an era of exponential growth in information data, recommendation systems have become a key technology to help users find content of interest from massive amounts of information [1-2]. Traditional recommendation algorithms such as collaborative filtering, content-based recommendation, and matrix decomposition methods have solved the problem of information overload to a certain extent. However, with the continuous growth of data scale and the diversification of user needs, such methods have gradually exposed their limitations [3-4]. In other words, traditional recommendation algorithms have difficulty processing complex multi-relational data, and when processing distributed and heterogeneous data, they do not adequately handle data consistency and privacy protection, resulting in unsatisfactory recommendation results. To improve the performance of recommendation systems, researchers have begun to introduce emerging technologies such as Knowledge Graphs (KGs) and Graph Neural Networks (GNNs) [5]. KG is a structured graph data that can effectively represent entities and their relationships, containing rich semantic information. By applying KG to recommendation systems, the accuracy and interpretability of recommendations can be significantly improved. Wang et al. proposed a KG navigation visualization query system, redefining the minimum operating unit. This study abstracted the conceptual hierarchy in the domain from the original KG and provided a series of operators and interactive graphical user interfaces to capture user query intentions and guide users in exploring KG patterns [6]. Huang et al. proposed a multi-relation KG completion method for local information fusion. This method connected entities and their adjacency relationships, modeled the direction of relationships using different weights, and applied attention mechanisms to mine local information between entity nodes. This method performed well on multi-benchmark datasets and medical KG datasets [7]. Zhang et al. proposed a hierarchical perception pairing relationship vector KG embedding model. This model is designed to capture relationship patterns, multiple relationship types, and hierarchical features in KGs through the use of paired relationship vectors and angle coordinates. This approach allows for the effective modeling of the diverse characteristics of head and tail entities. This model performed well in link prediction tasks on multiple datasets [8]. Liang et al. proposed a deep relational graph model that maximizes information to address the problem of KG embedding models only focusing on semantic information and ignoring graph structure information. This model used two adaptive relationship graph attention network encoders to solve the disconnection of KG, with faster convergence speed and better predictive performance [9].

GNN, as a deep learning method for processing graph-structured data, can capture complex graph relationships and has demonstrated its powerful feature learning ability in multiple fields. Wang et al. proposed an efficient GNN structure search algorithm, which designs a new federated evolutionary optimization strategy, fully considers the GNN architecture preferences of each client and applies a super network to accelerate model evaluation. This algorithm could recommend GNN models with excellent performance in a short period [10]. Lin et al. developed a framework that can map GNN training workloads to platforms. It optimized data paths and storage organization through advanced application programming interfaces, software, and accelerator generators to achieve efficient GNN training. This framework achieved 27.21 times bandwidth efficiency and 4.26 times acceleration [11]. Wang et al. proposed an encoding neighbor sampling framework by combining encoding techniques with GNN to address the issue of high data communication overhead in GNN training. This framework saved an average of 40.6%, 35.5%, and 16.5% in communication overhead, and reduced runtime by 12.1%, 17.0%, and 10.0% [12]. He et al. proposed a fusion algorithm that combines GNN with Transformer convolutional layers. This algorithm outperformed traditional models such as support vector machines and random forests in battery performance prediction, with an R2 value of 0.82 and a mean square error of 0.3161. This algorithm could deeply understand the crystal structure of batteries and pave the way for more complex and durable battery systems [13]. Finally, the research summarizes the research areas, indicator testing results, and limitations of the literature review above, as shown in Table 1.

Table 1: Literature summary table

Authors	Algorithms/Methods	Key Results	Limitations
Wang et al. [6]	KG Navigation	Improved query efficiency, reducing query time by 22%, and enhanced user satisfaction by 15%.	Limited scalability with large datasets.
Huang et al. [7]	Multi-Relational KG Completion with Local Information Fusion	Achieved 94.7% accuracy and 91.5% precision in multi-benchmark datasets, with a 12% improvement in KG completion tasks.	High computational complexity limits real-time application.
Zhang et al. [8]	HPRE	89.3% accuracy in link prediction, with a 10.5% increase in performance over baseline models	Reduced generalization capability across different KG types.
Liang et al. [9]	DRGI	Achieved 92.4% accuracy and 88.6% F1 score, with 9.2% faster convergence	Potential overfitting on small datasets.
Wang et al. [10]	FL-AGNNS	Increased model accuracy by 8.7%, with a reduction in memory usage by 12.5% during training.	High memory consumption during model training impacts scalability.
Lin et al. [11]	Hit GNN	27.21 times bandwidth efficiency improvement and 4.26 times acceleration	Limited application scope, focused mainly on specific GNN workloads.
Wang et al. [12]	CNS	Reduced communication overhead by 16.5% and runtime by 10.0%. Increased throughput by 11.2%	Requires fine-tuning for different network topologies.
He et al. [13]	Transformer-GNN and Basic-GNN	Achieved 0.82 R2 value and 0.3161 MSE	Limited cross-domain applicability.

Combined with Table 1, despite the extensive research conducted by numerous scholars on KG and

GNN, there are still issues with the accuracy and diversity of recommendation results, low computational efficiency, and high memory consumption. Given this, this study innovatively combines the Knowledge Graph Convolutional Network (KGCN) based on GNN with the Ripple Network (RippleNet) based on KG, and proposes a recommendation algorithm that combines KGCN and RippleNet called KGCNR. The Ripple Propagation Mechanism (RPM) of KG is an effective method for capturing the potential interest relationships of users. When combined with the ability of GNNs to learn user and item features, this approach has the potential to enhance the overall performance of the system. The model aims to ensure recommendation accuracy and diversity while considering issues of computing resources and efficiency, providing a feasible solution for large-scale applications.

2 Methods and materials

In response to the challenges of data sparsity and relationship complexity in existing recommendation algorithms, this study first introduces the basic framework of KGNN from the aspect of user item interaction. Secondly, it is further optimized by combining with RippleNet. This paper develops a fusion recommendation algorithm aimed at improving the accuracy and diversity of recommendation systems, thereby effectively addressing the increasingly complex demands in recommendation systems.

2.1 Recommendation algorithm based on KG and GNN

Google officially introduced the concept of KG in 2012, with the objective of achieving a more intelligent search engine and started to popularize it in academia and the industry after 2013 [14]. KG can aggregate information, data, and connectivity into knowledge, making information resources easier to calculate, understand, and evaluate, thus forming a semantic knowledge base. Essentially, it is a heterogeneous network topology structure with semantic information, which processes, integrates, and transforms complex data into a simple and clear set of (entities, relationships, entities) triplets, known as $\{h,r,t\}$ [15]. The construction process of KG is shown in Figure 1.

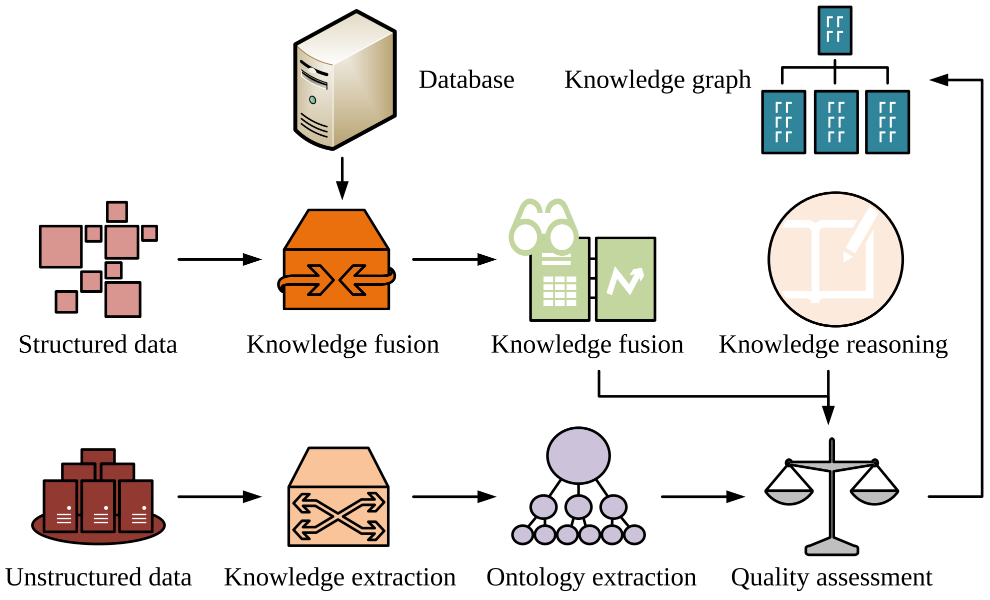


Figure 1: The process of constructing KG

As shown in Figure 1, the construction of KG mainly includes three stages: information extraction, knowledge fusion, and knowledge processing. In the information extraction stage, data are collected from various sources, including structured databases, unstructured text, and other multimedia sources. Subsequently, the entities, entity attributes, and the relationships between these entities are identified and extracted. After extraction, the collected information is integrated to form a coherent KG. This stage involves resolving inconsistencies, eliminating duplicate information, and merging data from different sources. Its goal is to create a unified representation of knowledge, minimize contradictions, and clarify ambiguities. The final stage involves processing the integrated knowledge to improve its quality and usability. This includes quality assessment to ensure that the knowledge meets predefined standards before being added to the knowledge base. Furthermore, the processing stage may entail the conversion of knowledge into a format that is compatible with a range of applications.

Due to the excellent performance of GNN in node classification, KG is often introduced into GNN-based recommendation systems [16]. GNN can be divided into two categories, namely Graph Convolutional Networks (GCN) and Message Passing Networks. GCN originates from convolutional operations in deep neural networks, which can effectively capture local information to form better representations. The expression for the k-th layer of GCN is equation (1) [17].

$$X^{(k)} = \phi_k(\tilde{L}X^{(k-1)}W^{(k)}) \quad (1)$$

In equation (1), ϕ_k represents the activation function. $X^{(k)}$ and $X^{(k-1)}$ represent layers k and $k-1$, respectively. $W^{(k)}$ represents the feature transformation matrix. $\tilde{L}$ represents the normalized Laplacian matrix. GNN based on message passing also follows the rules of old message passing algorithms, representing shared functions through GNN. KGCN is an end-to-end framework that effectively captures the correlation between items by mining relevant attributes on KG and combining Neighborhood Information (NI) with bias when calculating the representation of a given entity. The receptive field is extended to encompass multiple hops to simulate high-order NI and to facilitate the capture of potential user interests. Therefore, this study introduces KGCN and utilizes the rich semantic information provided by KG to handle the issue of data sparsity while enhancing the interpretability of the system and capturing high-order relationships. The process of KGCN is shown in Figure 2.

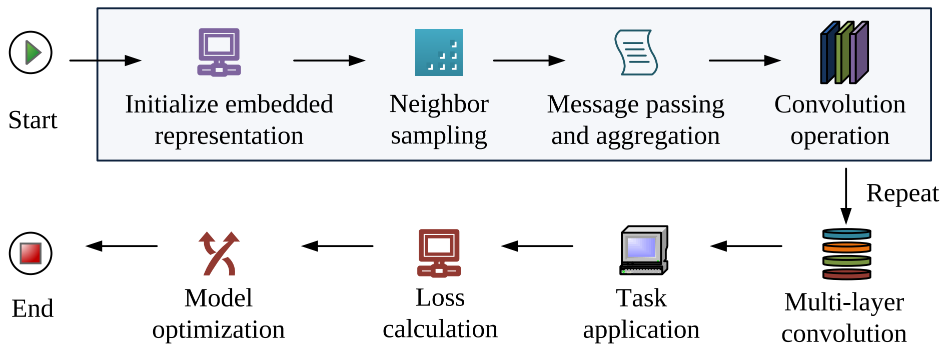


Figure 2: The process of KGCN

As shown in Figure 2, KGCN first initializes the embedding representation of each entity and relationship in the KG. These embeddings are low-dimensional vector representations that capture the basic characteristics of entities and relationships, making them suitable for further processing by convolutional networks. After initialization, a fixed number of neighbors are sampled for each entity. This step is intended to mitigate the computational burden by constraining the number of connections that are considered for each entity. Sampling enables the model to concentrate on the most pertinent or significant relationships, while disregarding those of lesser importance. The sampled neighbor information is then aggregated using a defined aggregation function. This function combines information from neighboring entities to enrich the representation of the target entity. Aggregation can be performed in a number of ways, including the addition, averaging, or application of more sophisticated neural operations. After aggregating neighbor information, the current entity representation undergoes a linear transformation followed by a nonlinear activation. This process refines the representation of the entity, enabling the model to capture more complex patterns and interactions in the data. The above steps are repeated and multiple layers of convolution are stacked. The final entity representation is then applied to a specific task, and a loss function is defined based on the specific task. Finally, an optimization algorithm is used to minimize the loss function and update the model parameters. In a single KGCN layer, the score calculation for users and relationships is equation (2) [18].

$$\pi_r^u = g(u,r) \quad (2)$$

In equation (2), u and r respectively represent users and relationships, while π_r^u is the importance of r to u. Subsequently, the linear combination calculation of project v is equation (3).

$$v_{S(v)}^u = \sum_{e \in S(v)} \tilde{\pi}_{r_{v,e}}^u e \quad (3)$$

In equation (3), $\tilde{\pi}_{r_{v,e}}^u$ is the normalized user relationship score. e is the entity, and $S(v)$ represents the entity's single-layer receptive field. The summation aggregator for aggregating entity v and neighborhood representation $v_{S(v)}^u$ into a single vector is equation (4).

$$agg_{sum} = \sigma(W \cdot (v + v_{S(v)}^u) + b) \quad (4)$$

In equation (4), W is the linear transformation weight. b means the deviation, and σ is the nonlinear function. v represents the entity, and $v_{S(v)}^u$ represents the neighborhood. Through the aggregation step, the project representation is bound to its neighbors. KGCN has a strategy to extend representations from single-layer structures to multi-layer structures. This multi-layered design can capture deeper user preferences, making recommendation results more personalized and accurate. The framework is shown in Figure 3.

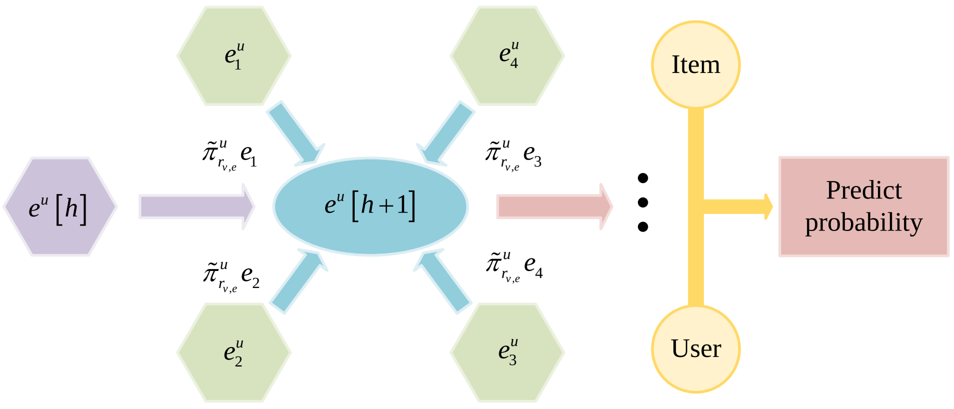

e_1^u
e_4^u
e_2^u
e_3^u
$e^u[h]$
$e^u[h+1]$
$\tilde{\pi}_{r_{v,e}}^u e_1$
$\tilde{\pi}_{r_{v,e}}^u e_2$
$\tilde{\pi}_{r_{v,e}}^u e_3$
$\tilde{\pi}_{r_{v,e}}^u e_4$
Item
User
Predict probability

Figure 3: The architecture of KGCN

In Figure 3, the purple node represents the entity of the given node, the green node is the neighborhood, and the blue node is the next iteration. The KGCN algorithm mixes the entity representation and domain representation of a given node in one iteration, while forming the representation for the next iteration. KCGN integrates KG into recommendation systems, providing users with clearer and more reliable recommendation reasons.

2.2 Construction of a fusion model based on KG and GNN

However, KGCN mainly focuses on the project side in KG, and has some shortcomings in user side modeling. The construction of the neighborhood structure is based on a unified sampling strategy of hyperparameters, whereby the probability of each node being selected is equal, and the propagation process of historical interaction information on the user end is ignored. On the contrary, RippleNet adopts a different strategy, which utilizes the RPM to enrich the representation of user feature vectors. RippleNet is also an algorithm that combines KG feature vectors with recommendation systems. It introduces KG embedding method into recommendation through preference propagation, which can expand user interests and discover potential preferences with KG iteration, and has good recommendation effect and model interpretability. Therefore, this study combines KGCN with RippleNet and proposes a recommendation model KGCNR that combines KG-based ripple propagation with GNN. The model architecture of KGCNR is shown in Figure 4.

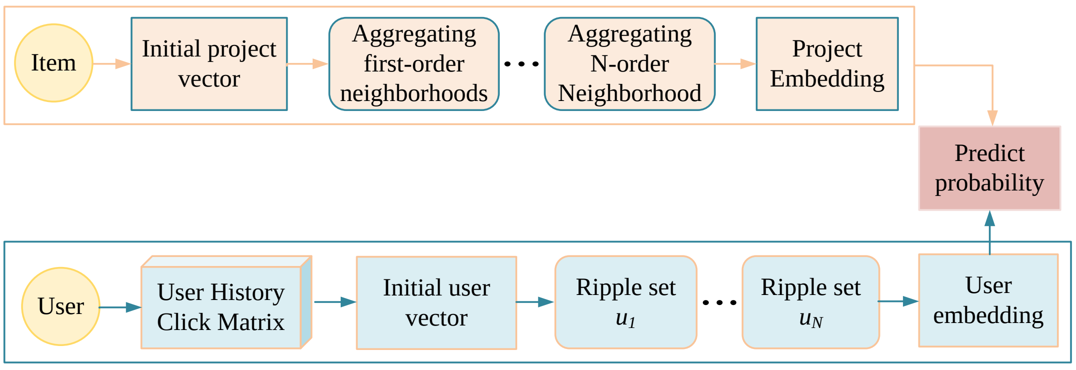


Figure 4: The model architecture of KGCNR

In Figure 4, Step 1 is to obtain potential preference items through historical data. Secondly, after constructing KG, all vectors are obtained through RippleNet. Finally, the user embedding vectors are weighted and summed to generate higher-order user feature representations to enrich the user profile. Step 2 is to capture the NI of the target object based on its corresponding entity in KG through the receptive field mechanism, and convert the relationships between entities into weights through calculation. Multiple relationship weights are further sampled and aggregated to obtain higher-order term representations. Step 3 is to input high-order user and item feature representations into the prediction function to generate recommendation results. KGCNR enriches user feature representation through ripple network propagation. The messaging mechanism of GNN on the item side is utilized to aggregate NI and enrich the feature representation of the item. It aims to maximize the utilization of information in KG to improve the performance of recommendation systems. The process of generating high-order user feature vectors is shown in Figure 5.

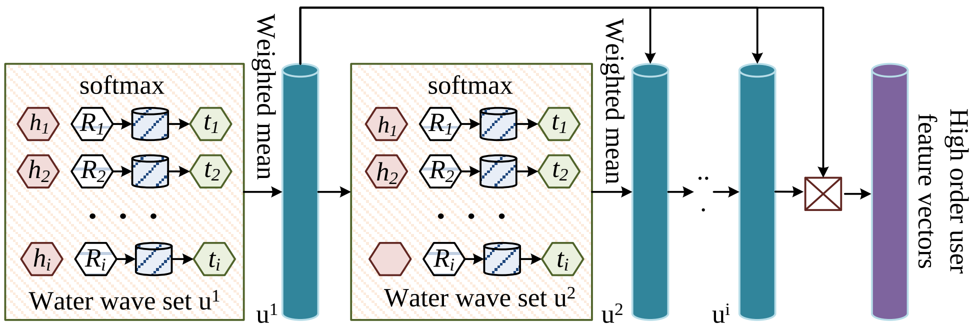


Figure 5: Generation of high-order user feature vectors

In Figure 5, using matrix $User-Item$, $Item$ with historical interaction records with $User$ is selected and corresponding to KG to form the initial set. Subsequently, information is aggregated layer by layer for each entity within it. The first tail entity in the propagation is used as the head entity of the next layer to find the corresponding tail entity of this layer. Therefore, the calculation of the user's h-hop entity set is equation (5).

$$\varepsilon_u^{(h)} = \left\{ t \middle| (h,r,t) \in G \quad and \quad h \in \varepsilon_u^{(h-1)} \right\} \tag{5}$$

In equation (5), u represents the user. $\varepsilon_u^{(h)}$ and $\varepsilon_u^{(h-1)}$ are the entity collections of user u's h and $h-1$ hops. (h,r,t) is a triplet, $(h,r) \to t$. G represents KG. Due to the different similarities between different relationships in the same project entity pair, relationship vectors are added to the calculation, as shown in equation (6).

$$\begin{aligned} p_i &= softmax(h_i r_i v^T) \\ &= \frac{\exp(h_i r_i v^T)}{\sum_{(h,r,t)\in S_u^{(h)}} \exp(h_i r_i v^T)} \end{aligned} \tag{6}$$

In equation (6), h_i, r_i, and v^T are the vectors of the i-th (h,r,t). $S_u^{(h)}$ represents the triplet set corresponding to the entity set of the h hop. Subsequently, the user vector features are calculated based on the probability value p_i, as shown in equation (7).

$$u^{(h)} = \sum\nolimits_{(h_i,r_i,t_i)\in S_u^h} p_i t_i \tag{7}$$

In equation (7), $u^{(h)}$ is the user feature vector of the h-th layer obtained by weighted summation. $S_u^{(h)}$ represents the corresponding set of triples. t_i is the tail entity embedding representation in the i-th triplet. p_i represents the probability value. Subsequently, a weighted summation strategy is adopted when constructing higher-order representations of users. Specifically, for each layer, the user feature vectors obtained through RPM are assigned corresponding weights based on the number of propagation layers. These weights are allocated based on the number of propagation layers to reflect the importance of different hierarchical features in the final user representation. Finally, all weighted user feature vectors are summed to generate a high-order representation of the user, as shown in equation (8).

$$u_w = \sum_{h=1}^{L} w_h u^{(h)} \tag{8}$$

In equation (8), $u^{(h)}$ represents the user feature vector of the h layer obtained by weighted summation. u_w is the final higher-order representation of the user, and w_h is the weight assigned based on the number of propagation layers. L represents the overall propagation layers. In the calculation of high-order item feature vectors, in KGCN, the relationships between entities are transformed into weights, that is, the degree to which relationships affect user behavior preferences. The triplet set of items obtained from the entity set during the aggregation process is equation (9).

$$\begin{cases} \varepsilon_v^{(h)} = \left\{ t \middle| (h,r,t) \in Gandh \in \varepsilon_v^{(h-1)} \right\} \\ S_v^{(h)} = \left\{ (h,r,t) \middle| (h,r,t) \in Gandh \in S_v^{(h-1)} \right\} \end{cases} \tag{9}$$

In equation (9), $\varepsilon_v^{(h)}$ denotes the set of entity entities of the item, and $S_v^{(h)}$ is the set of triples of the item. Therefore, the transformation of the relationship vector towards weights is shown in Figure 6.

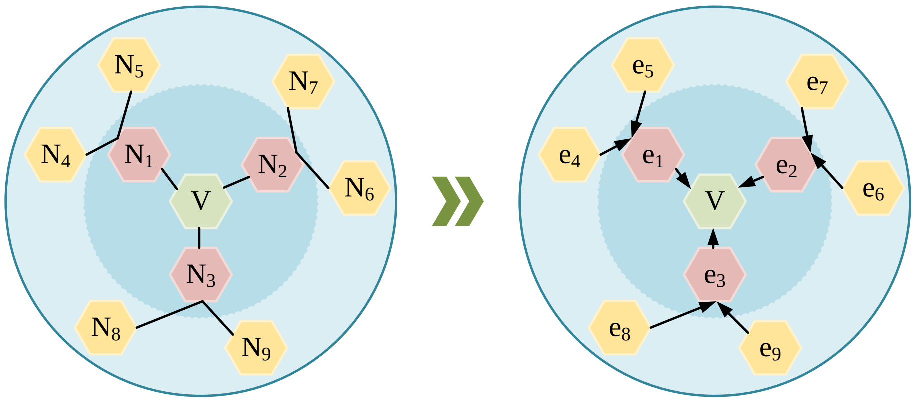


Figure 6: Relationship vector transformation process

In Figure 6, v is the target item. After determining v, its corresponding entities are obtained and an initial set of entities is constructed. Then, with each entity as the center, a strategy from outside to inside is adopted to gradually aggregate its NI, and finally obtain the transformed weights. Subsequently, the obtained weights will be normalized as shown in equation (10).

$$\tilde{w}_{R_i}^{U} = softmax_j\left(w_{R_i}^{U}\right) = \frac{\exp(w_{R_i}^{U})}{\sum_{j\in N(V)}\exp(w_{R_i}^{U})} \tag{10}$$

In equation (10), $w_{R_i}^{U}$ represents the weight. $\tilde{w}_{R_i}^{U}$ represents the weight normalized by $softmax$. V is the node, and $N_{(v)}$ is the first-order neighbor set of node V. Subsequently, the feature vector representation is weighted and summed with the neighboring entity vectors, followed by a fully connected layer calculation, as shown in equation (11).

$$v = \sigma(W \cdot agg(v_{N(V)}^{u}, e) + b) \tag{11}$$

In equation (11), W is the linear transformation weight. b means the bias term. σ is the activation function Relu. $agg(v_{N(V)}^{u}, e)$ is the message aggregation of the item, and the final item feature representation is obtained by substituting the sum aggregation method. Then, the above two terms are input into the prediction function, as shown in equation (12).

$$\hat{y}_{uv} = \sigma'(u_w^T v) \tag{12}$$

In equation (12), $\hat{y}_{uv}$ is the user's preference probability for the item, and σ' represents the Sigmoid prediction function. $u_w^T v$ represents the inner product of the user vector and the item vector. Finally, a negative sample distribution strategy is adopted to train the model to improve training efficiency, and its loss function is equation (13).

$$L = \sum_{u=U}\Big(\sum_{v:y_{uv}=1} J(y_{uv}, \hat{y}_{uv}) - \sum_{i=1}^{T_u} E_{v_i \sim P(v_i)} J(y_{uv_i}, \hat{y}_{uv_i})\Big) + \lambda\|F\|_2^2 \tag{13}$$

In equation (13), $J(y_{uv_i}, \hat{y}_{uv_i})$ is the cross-entropy loss. P and T_u are negative sample strategy and quantity. $\lambda\|F\|_2^2$ is a regularization term to prevent overfitting of the model. U represents the set of all users. $E_{v_i \sim P(v_i)}$ represents the expectation of negative samples $P(v_i)$ sampled from distribution. v_i represents the average loss of all possible negative samples. F represents the parameter matrix of the model. Regularization term $\|F\|_2^2$ constrains the parameters through L2 norm to prevent the model's parameters from being too large while maintaining the model's generalization ability.

3 Results

To verify the performance of KGCNR, this study first established a suitable experimental environment and conducted basic tests on the parameters and performance of the model. Secondly, to verify its effectiveness in practical scenarios, tests were conducted using real user behavior data to evaluate its application effectiveness in actual recommendation systems.

3.1 Performance comparison test of KGCNR

To evaluate the effectiveness of recommendation models in different fields, three commonly used datasets for testing recommendation performance were selected: the movie dataset MovieLens-1m (Movie), the music dataset Last-FM (Music), and the book dataset Book-Crossing (Book). The three were segmented into training and testing sets in an 8:2 ratio. Firstly, using the Area Under the Curve (AUC) value as an indicator, three hyper-parameters, namely the number of propagation hops h on the user end, the gross of propagation layers L in the receiving domain on the item end, and the number of neighbor samples N, were adjusted through experiments. The results are shown in Figure 7.

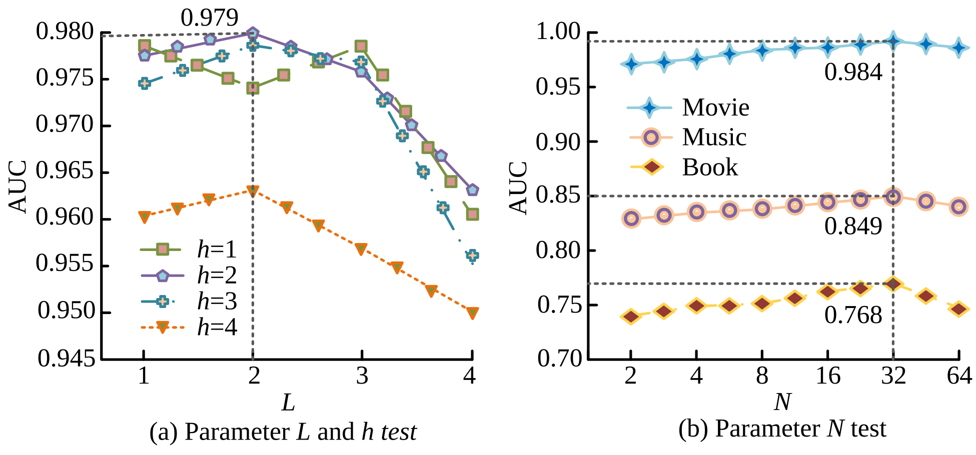


Figure 7: Results of parameter adjustment

Figure 7 (a) shows the AUC values using different h and L values on the MovieLens-1m test set. Figure 7 (b) shows the AUC test results using different N values on the Movie, Music, and Book datasets. In a recommendation system, the larger the AUC value, the higher the proportion of items that users are preferred, indicating a positive sample in the recommendation results. In Figure 7 (a), when the h is 2 and the L is 2, the model achieves the optimal AUC value of 0.979. When the h and L are 4, the recommendation effect is the worst, because in RippleNet, features with lower relevant information are introduced when h is higher, resulting in a decrease in recommendation effect. In Figure 7 (b), when the N value is set to 32, the AUC values on the three datasets are 0.984, 0.849, and 0.768, indicating the best recommendation performance of the model. However, when N is 64, the recommendation quality shows a downward trend because when there are too many neighboring samples in KG, the model may overfit. Therefore, the values of h, L, and N parameters are 2, 2, and 32. Secondly, as KGCNR is a combination of KGCN and RippleNet, ablation tests are conducted on the MovieLens-1m dataset to investigate the effect of each module on the model, as shown in Figure 8.

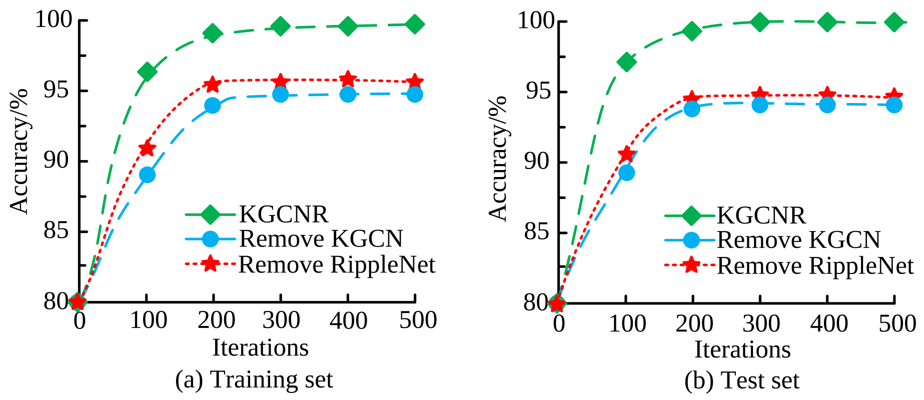


Figure 8: Ablation test results

Figures 8 (a) and (b) show the ablation test results on the training and testing sets for KGCNR, KGCNR with KGCN module removed, and KGCNR with RippleNet module removed. When iterating 500 times, the accuracy of the three is 99.73%, 94.98%, and 96.04% in Figure 8 (a), and 99.05%, 93.93%, and 94.81% in Figure 8 (b). The accuracy of KGCNR without KGCN module is slightly lower than that of KGCNR without RippleNet module, due to the significant enhancement effect of RippleNet's ripple characteristics on user representation. Therefore, both the KGCN and RippleNet modules have a certain degree of positive effect on the final KGCNR model. Finally, Neural Matrix Factorization (NeuMF), Knowledge Graph Attention Network (KGAT), and KGCN are introduced as comparison models for comprehensive performance comparison testing with the KGCNR model, as listed in Table 2.

Table 2: Comprehensive performance comparison test results

Data set	Index	Model			
		NeuMF	KGAT	KGCN	KGCNR
Movie	P	0.937	0.948	0.951	0.957
	R	0.917	0.918	0.926	0.928
	F1	0.927	0.933	0.938	0.942
	AUC	0.914	0.962	0.961	0.982
	NDCG	0.920	0.960	0.940	0.970
Music	P	0.801	0.829	0.798	0.901
	R	0.797	0.824	0.801	0.897
	F1	0.799	0.826	0.799	0.899
	AUC	0.795	0.819	0.796	0.849
	NDCG	0.730	0.820	0.780	0.870
Book	P	0.684	0.721	0.667	0.804
	R	0.702	0.734	0.698	0.795
	F1	0.693	0.727	0.682	0.799
	AUC	0.721	0.693	0.685	0.797
	NDCG	0.710	0.750	0.720	0.810

Table 2 shows the comprehensive performance results of NeuMF, KGAT, KGCN, and KGCNR models on different datasets. Normalized Discounted Cumulative Gain (NDCG) is used to represent the ranking quality of recommendations, with values approaching 1 indicating better ranking quality. In the Movie, Music, and Book datasets, the NDCG values of KGCNR are 0.970, 0.870, and 0.810. Compared to KGCN, KGCNR has shown varying degrees of progress, confirming that the improvement of KGCNR has a good effect. NeuMF is a non-KG recommendation model, therefore the numerical values in comparative experiments are lower than other models that use KG, which also verifies the effectiveness of using KG to extract feature vectors in recommendation systems.

3.2 Practical application performance testing of recommendation model based on KGCNR

Due to the best performance of each model in movie recommendation, the movie recommendation direction is also adopted in practical applications. This study randomly selects 400 college students aged 18 to 25 from Beijing and crawls their recent year's movie watching records. The final recommended number of movies is set to 5, 10, 15, 20, 25, and 30 to test the model's recommendation matching, as shown in Figure 9.

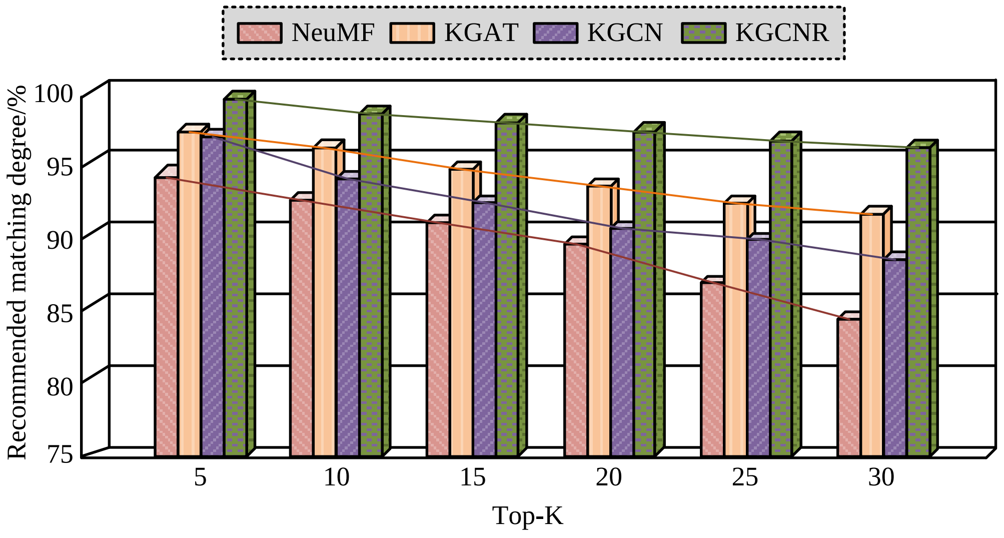
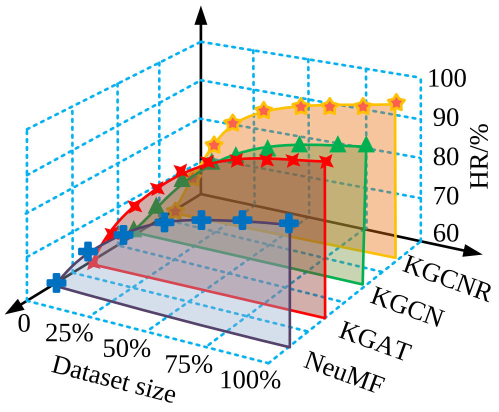
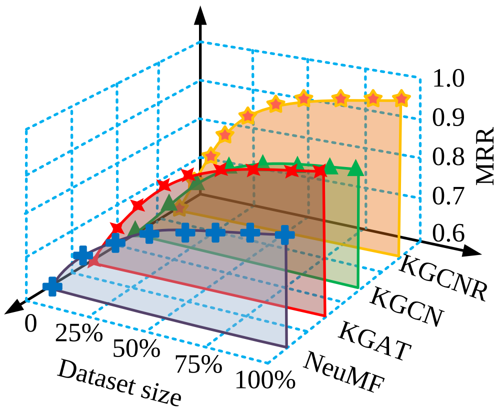


Figure 9: Top-K recommendation matching test

Figure 9 shows the final recommendation matching degree of NeuMF, KGAT, KGCN, and KGCNR models at Top-K values of 5, 10, 15, 20, 25, and 30. As the number of recommended movies increases, the recommendation matching rates of each model gradually decrease. When the final number of recommendations is 5, 10, 15, 20, 25, and 30, the recommendation matching degrees of the KGCNR model are 99.2%, 98.3%, 97.4%, 96.5%, 95.8%, and 95.2%, respectively. When the final number of recommendations is 30, the recommendation matching degrees of NeuMF, KGAT, KGCN, and KGCNR models are 83.7%, 91.5%, 87.3%, and 95.2%. Subsequently, using Hit Ratio (HR) and Mean Reciprocal Rank (MRR) as indicators, the Top-K is set to 20. The dataset is further tested by dividing it into four parts based on 25%, 50%, 75%, and 100%, as shown in Figure 10.



(a) HR Test



(b) MRR test

Figure 10: HR and MRR test results

Figures 10 (a) and (b) show the HR and MRR test results for four models. HR measures the accuracy of recommendation results by the proportion of items that users actually click on in the recommendation list, which is an objective indicator of the performance of recommendation systems. A higher MRR value indicates that users can find more interesting items in the recommendation results generated by the system. When the dataset size is 100%, the HR values of NeuMF, KGAT, KGCN, and KGCNR are 82.7%, 93.4%, 91.8%, and 98.7%, and the MRR values are 0.82, 0.89, 0.84, and 0.96. Finally, this study expands the experimental scope by selecting small, medium, and large movie datasets to test the Runtime and memory consumption on datasets of different scales. The small-scale dataset contains 10,000 user records and 1,000 movies, the medium-sized dataset contains 50,000 user records and 5,000 movies, and the large-scale dataset contains 100,000 user records and 10,000 movies. Table 3 lists the test results.

Table 3: Time and memory usage tests on data sets of different sizes

Model	Small-scale datasets		Medium-sized datasets		Large-scale datasets	
	Runtime/s	Memory size/MB	Runtime/s	Memory size/MB	Runtime/s	Memory size/MB
NeuMF	2.8	146	12.5	528	24.9	1057
KGAT	3.1	152	12.2	552	25.8	1186
KGCN	2.3	137	11.1	514	23.4	1024
KGCNR	2.1	117	10.4	480	21.5	958

Table 3 shows the runtime and memory consumption tests of NeuMF, KGAT, KGCN, and KGCNR on small, medium, and large digital movie datasets, respectively. The recommended runtime of the KGCNR model on three datasets is 2.1s, 10.4s, and 21.5s, and the memory consumed is 117MB, 480MB, and 958MB. The running time of KGCNR in the comparative models is optimal, and the memory consumption is minimal, proving its scalability and computational efficiency, as well as its good robustness. Finally, to further test the practicality and scalability of the model, the test results of various indicators of the model at different time intervals and data transmission rates are shown in Table 4.

Table 4: Comparison of different models in real time data stream processing

Model	Time Interval/min	Incoming Data Rate /Records/s	Resource Usage/%	Memory Consumption /MB	Processing Latency/ms
KGCNR	10-20	5,000	47.83	772.54	34.67
	30-40	15,000	71.96	1283.42	69.54
	50-60	25,000	91.67	1789.31	119.78
NeuMF	10-20	5,000	55.29	849.72	49.38
	30-40	15,000	79.45	1448.69	104.57
	50-60	25,000	97.83	1998.47	159.65
KGAT	10-20	5,000	51.67	819.54	44.32
	30-40	15,000	76.29	1349.81	89.73
	50-60	25,000	94.72	1897.56	149.87
KGCN	10-20	5,000	53.16	837.45	47.82
	30-40	15,000	77.34	1378.63	99.34
	50-60	25,000	95.68	1918.47	158.79

In Table 4, the resource consumption rate, memory consumption, and processing delay test results of four models under different time intervals and data transmission volumes are presented. KGCNR exhibits lower resource consumption and memory utilization at different time intervals and data flow rates. Especially in the 50–60-minute interval with the highest data flow rate, its resource consumption rate is 91.67%, significantly lower than NeuMF's 97.83% and KGAT's 94.72%. The processing delay remains within 120 ms, while NeuMF and KGCN exhibit delays of 159.65 ms and 158.79 ms, respectively, at the same data flow rate. This suggests that KGCNR can process substantial quantities of real-time data with greater efficiency, reducing system resource utilization and latency while maintaining high accuracy, rendering it a more suitable choice for deployment in large-scale real-time applications that necessitate rapid response times.

4 Discussion

To improve the system performance, this study proposed a novel recommendation algorithm KGCNR that combines KG ripple propagation and GNN. Chen et al. [18] proposed a double-layer GCN recommendation model, LighterKGCN, using a mixed aggregation function. Similar to KGCNR, it had a significant improvement in dealing with data sparsity issues. However, the AUC value of LighterKGCN only increased by 0.52% and 0.67% on the Movie and Music datasets. The AUC values of KGCNR on the Movie, Music, and Book datasets were 0.982, 0.849, and 0.810, which increased by 2.19%, 11.54%, and 12.5%. This was because the KGCNR model combined KG and GNN, especially by fusing the relationship information in KG, enhancing the performance. The KGCNR model can effectively utilize the global semantic information in the KG to help identify higher-order relationships. This not only enhances the recommendation ability in sparse data environments, but also improves the model's accuracy in capturing diverse user interests. In addition, KGCNR integrates multi-level graph convolution operations to enable the model to more accurately represent the features of users and items, thereby improving the overall performance of the recommendation system. On the sparse Book-Crossing dataset, the P, R, F1, and NDGG values of the model were 0.804, 0.795, 0.799, and 0.810, respectively.

The KGCN-LSTM model proposed by Chen et al. [17] combined GCN and LSTM. Although KGCN-LSTM performed well in processing temporal data, its application in recommendation systems was limited. KGCNR provided stronger recommendation capabilities by integrating GCN and KG, especially when dealing with non-temporal user behavior data. In the Top-K test, when the final number of recommendations was 5, 10, 15, 20, 25, and 30, the recommendation matching degree of KGCNR was 99.2%, 98.3%, 97.4%, 96.5%, 95.8%, and 95.2%, which were better than the other models. KGCNR enhanced recommendation accuracy through KG, providing a more targeted solution. In addition, in HR and MRR value tests on datasets of different sizes, KGCNR achieved the highest values on datasets of 25%, 50%, 75%, and 100%. In practical applications, the running time of KGCNR on small, medium, and large datasets was 2.1s, 10.4s, and 21.5s, and the memory consumption was 117MB, 480MB, and 958MB. In summary, KGCNR could effectively reduce computational complexity and data redundancy while ensuring high accuracy. In addition, KGCNR performed better than other models in real-time data stream processing. At a data flow rate of 25,000 Records/s within a 50–60-minute time interval, the resource consumption rate of KGCNR was 91.67%, significantly lower than NeuMF's 97.83% and KGAT's 94.72%. This was mainly due to the optimization of the model in graph convolution operations, which enabled KGCNR to maintain efficient feature extraction capabilities and lower computational burden when processing large-scale data.

In summary, this study proposes an efficient and accurate recommendation system fusion model by combining KG's RPM and GNN. The KGCNR model successfully integrates rich semantic information from KGs with the deep learning capabilities of GCNs, providing more personalized and accurate recommendation results when processing complex multi-relational data. The excellent performance of KGCNR in large-scale real-time data processing provides an efficient and low resource consumption solution for recommendation systems in practical applications. In addition, by introducing RippleNet's RPM, KGCNR not only improves the accuracy of recommendations, but also enhances the interpretability of the model, making it easier for users to understand the reasons behind the recommendation results. It not only provides new ideas and technical means for the development of future recommendation systems, but also has a positive impact on improving the accuracy, diversity, and robustness of systems.

5 Conclusion

To improve the operational efficiency of recommendation systems, this study proposed a KGCNR fusion recommendation model by combining KG's RPM and GNN. The results showed that the model not only achieved excellent test results in NDCG, AUC, HR, MRR and other indicators, but also confirmed its high recommendation quality in Top-K and data sparsity testing. RPM fully utilized the global information of KG by spreading user interest information, while GNN further enhanced the expression ability of these interest patterns, thereby achieving more efficient recommendation results. In summary, the KGCNR model not only performs well in performance testing but also achieves excellent recommendation results in practical applications. Nevertheless, despite the favorable outcomes in terms of recommendation accuracy, a significant hurdle persists for deep learning models, including KGCNR, in that they are inherently opaque. Further research could concentrate on the development of techniques to render the decision-making process of KGCNR more comprehensible to end users. This involves integrating explainable AI (XAI) techniques to enable users to understand why certain recommendations are made, potentially increasing trust and adoption of the system. At the same time, the current research focuses on specific domains such as movies, music, and books. Future research can explore the generalization capabilities of KGCNR to other domains such as e-commerce, healthcare, and social media. It is recommended that the performance of the model be studied in these different environments to gain further insights and verify its broad applicability.

6 Funding

The research is supported by Social Sciences Fund projects of Henan Universities "Study on the Path to Enhance the Data Analysis Ability of Public Security under the Requirements of New Quality Public Security Combat Capability" (673); Scientific Research Fund Project of Anyang Institute of Technology "Research on a Predictive Model for Psychological Crisis among College Students Caused by Social Pressure based on Big Data Analysis" (YPY2021009).

References

[1] p. p. groumpos, "a critical historic overview of artificial intelligence: issues, challenges, opportunities, and threats," artificial intelligence and applications, vol. 1, no. 4, pp. 197-213, 2023. https://doi.org/10.47852/bonviewaia3202689

[2] g. chornous, and t. e. lem, "developing hybrid recommendation systems: ukrainian dimension," access journal, vol. 3, no. 2, pp. 89-106, 2022. https://doi.org/10.46656/access.2022.3.2(1)

[3] s. hwang, and e. park, "movie recommendation systems using actor-based matrix computations in south korea," ieee transactions on computational social systems, vol. 9, no. 5, pp. 1387-1393, 2021. https://doi.org/10.1109/tcss.2021.3117885

[4] x. ma, "recommendation of sustainable economic learning course based on text vector model and support vector machine," journal of intelligent & fuzzy systems, vol. 40, no. 4, pp. 7135-7145, 2021. https://doi.org/10.3233/jifs-189542

[5] m. sharaf, e. hemdan, a. el-sayed, and n. el-bahnasawy, "a survey on recommendation systems for financial services," multimedia tools and applications, vol. 81, no. 12, pp. 16761-16781, 2022. https://doi.org/10.1007/s11042-022-12564-1

[6] x. wang, x. wang, z. li, and d. han, "kgnav: a knowledge graph navigational visual query system," proceedings of the vldb endowment, vol. 16, no. 12, pp. 3946-3949, 2023. https://doi.org/10.14778/3611540.3611592

[7] j. huang, t. lu, j. zhu, w. yu, and t. zhang, "multi-relational knowledge graph completion method with local information fusion," applied intelligence, vol. 52, no. 7, pp. 7985-7994, 2022. https://doi.org/10.1007/s10489-021-02876-4

[8] d. zhang, j. liu, d. liu, and g. li, "hpre: leveraging hierarchy-aware paired relation vectors for knowledge graph embedding," journal of intelligent & fuzzy systems, vol. 45, no. 4, pp. 5907-5926, 2023. https://doi.org/10.3233/jifs-230982

[9] s. liang, j. shao, d. zhang, j. zhang, and b. cui, "drgi: deep relational graph infomax for knowledge graph completion," ieee transactions on knowledge and data engineering, vol. 35, no. 3, pp. 2486-2499, 2021. https://doi.org/10.1109/tkde.2021.3110898

[10] c. wang, b. chen, g. li, and h. wang, "automated graph neural network search under federated learning framework," ieee transactions on knowledge and data engineering, vol. 35, no. 10, pp. 9959-9972, 2023. https://doi.org/10.1109/tkde.2023.3250264

[11] y. lin, b. zhang, and v. prasanna, "hitgnn: high-throughput gnn training framework on cpu plus multi-fpga heterogeneous platform," ieee transactions on parallel and distributed systems, vol. 35, no. 5, pp. 707-719, 2024. https://doi.org/10.48550/arxiv.2303.01568

[12] y. wang, t. guan, d. niu, q. zou, h. zheng, c. j. shi, and y. xie, "accelerating distributed gnn training by codes," ieee transactions on parallel and distributed systems, vol. 34, no. 9, pp. 2598-2614, 2023. https://doi.org/10.1109/tpds.2023.3295184

[13] x. he, y. chen, s. wang, and g. zhang, "employing graph neural networks for predicting electrode average voltages and screening high-voltage sodium cathode materials," acs applied materials & interfaces, vol. 16, no. 19, pp. 24494-24501, 2024. https://doi.org/10.1021/acsami.4c00624

[14] a. liu, d. zhang, y. wang, and x. xu, "knowledge graph with machine learning for product design," cirp annals, vol. 71, no. 1, pp. 117-120, 2022. https://doi.org/10.1016/j.cirp.2022.03.025

[15] c. wang, x. wang, and k. liu, "multimodal knowledge graph representation learning: a review," journal of computer applications, vol. 44, no. 1, pp. 2-16, 2024. https://doi.org/10.11772/j.issn.1001-9081.20230505 83

[16] d. tang, j. wang, r. chen, l. wang, w. yu, j. zhou, and k. li, "xgnn: boosting multi-gpu gnn training via global gnn memory store," proceedings of the vldb endowment, vol. 17, no. 5, pp. 1105-1118, 2024. https://doi.org/10.14778/3641204.3641219

[17] j. chen, d. fan, x. qian, and l. mei, "kgcn-lstm: a graph convolutional network considering knowledge fusion of point of interest for vehicle trajectory prediction," iet intelligent transport systems, vol. 17, no. 6, pp. 1087-1103, 2023. https://doi.org/10.1049/itr2.12341

[18] p. chen, j. zhao, and x. yu, "lighterkgcn: a recommender system model based on bi-layer graph convolutional networks," journal of internet technology, vol. 23, no. 3, pp. 621-629, 2022. https://doi.org/10.53106/160792642022052303020