2024 4th International Conference on Neural Networks, Information and Communication (NNICE) | 979-8-3503-9437-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/NNICE61279.2024.10499064

Enhancing Recommender Systems Performance using Knowledge Graph Embedding with Graph Neural Networks

Mingye Wang*
School of Automation Science and Electrical Engineering
Beihang University
Beijing, China
*Corresponding author: marcel0829@buaa.edu.cn

Xiaohui Hu
Institute of Software
Chinese Academy of Science
Beijing, China

Yao Du
School of Automation Science and Electrical Engineering
Beihang University
Beijing, China

***Abstract*—Recommender systems are a crucial component in personalized information retrieval, with their performance directly impacting user experience. However, traditional recommender systems often suffer from limitations in handling complex user-item interactions. Recently, knowledge graphs have emerged as valuable resources for enhancing recommendation effectiveness by capturing and utilizing a wide range of information. In this work, we propose a novel Auto-encoder-based structure integrated with Graph Neural Networks (GNN) layers for Knowledge Graph Embedding (KGE) tasks. This model effectively captures and leverages the rich information in knowledge graphs. Further, we combine this advanced GNN-based knowledge graph embedding model and a preference propagation mechanism inspired by RippleNet, to introduce a novel model, GraphRipple, aiming to enhance the recommendation performance. Extensive experiments on multiple datasets demonstrate the effectiveness of GraphRipple in improving recommendation quality.**

Keywords-recommender systems; knowledge graph; graph neural networks; knowledge graph embedding

I. INTRODUCTION

Recommender systems have become a critical tool in various online platforms, including e-commerce, social media, and content sharing platforms, to personalize user experience. They help users navigate through vast amounts of data by suggesting items of interest based on their previous interactions. However, traditional recommender systems often face challenges such as cold start problem, sparsity problem, and inability to handle complex user-item interactions [1].

Recently, knowledge graphs have been recognized as powerful tools that can alleviate these limitations. Knowledge graphs can represent and store a vast range of information, from simple user-item interactions to more complex relationships between items, leading to more accurate and meaningful recommendations. However, effectively leveraging the rich information in knowledge graphs for recommender systems remains a challenging task.

In this work, we propose a novel approach for knowledge graph embedding using an auto-encoder-based model that incorporates graph neural networks (GNNs) layers for feature extraction. This model is designed to capture and utilize the rich information in knowledge graphs effectively, thereby enhancing the performance of recommender systems.

A perfect embodiment of integrating knowledge graph embedding into recommender systems is RippleNet [2]. RippleNet uniquely models the propagation of user preferences on the knowledge graph by simulating the spread of 'ripples'. In this work, we introduce GraphRipple, a novel model that integrates our advanced GNN-based knowledge graph embedding model into this ripple propagation mechanism, aiming to enhance its recommendation performance by strengthening its knowledge graph embedding module.

Through extensive experiments on multiple datasets, we demonstrate that our approach significantly improves the recommendation quality compared to conventional methods. The details of our model and experimental results will be elaborated in the following sections.

The rest of the paper is organized as follows: Section 2 provides an overview of related work in the field of recommender systems and knowledge graph embeddings. Section 3 presents the methodology and details of our proposed approach. Section 4 describes the experimental setup and presents the results and analysis. Finally, Section 5 concludes the paper, highlighting the contributions of our work and discussing potential future research directions.

II. RELATED WORKS

In this section, we review the related works in three main areas: recommender systems, knowledge graph embedding, and graph neural networks.

A. Recommender Systems (RS)

Recommender systems are used in a variety of applications, from e-commerce to social networking, to provide personalized recommendations to users [3]. Traditional methods include collaborative filtering and content-based filtering [4], which have their own advantages and limitations. Recently, hybrid methods that combine multiple techniques have been proposed

to overcome these limitations [4]. Moreover, the use of deep learning in recommender systems has been a hot topic in recent years. For example, RippleNet is a notable deep-learning-based model that integrates knowledge graph embedding network into recommender systems by simulating the propagation of user preferences on the knowledge graph [2].

B. *Knowledge Graph Embedding (KGE)*

Knowledge graphs (KGs), which capture relationships between entities, have been utilized to enhance the performance of recommender systems [5]. Various methods have been proposed for knowledge graph embedding, such as TransE [6] and TransR [7]. Recently, some studies have begun to explore the use of auto-encoder-based structures to improve the performance of KGE [8, 9]. However, existing auto-encoder-based structures mainly focus on the features of the nodes themselves, without fully utilizing the connectivity of the graph.

C. *Graph Neural Networks (GNNs):*

GNNs have emerged as a powerful tool for dealing with graph-structured data. There have been various types of GNNs proposed, including Graph Convolutional Networks (GCN) [10], GraphSAGE [11] and Graph Attention Networks (GAT) [12]. GNNs have been applied in various fields, including physical systems [13], protein structure [14], and KGs [15].

III. METHODOLOGY

A. *Problem formulation*

In the context of recommender systems, we are given a set of users U, a set of items V, and a interaction matrix between users and items $Y = \{y_{uv} \mid u \in U, v \in V\}$. The matrix Y is defined as follow to represent implicit user feedback:

$$y_{uv} = \begin{cases} 1, & \text{if interaction (u,v) observed} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

In addition to the user-item interactions, we are also given a knowledge graph G, which consists of a set of entities E and a set of relations R. Each entity $e \in E$ can be an item or any other type of entity related to the items, such as director, lead actor, or screenwriter in a movie dataset. Each relation $r \in R$ indicates a specific type of relationship between two entities.

The goal of our GraphRipple model is to effectively leverage the information in the knowledge graph G to enhance the performance of the recommender system. Specifically, GraphRipple aims to generate powerful embeddings for the entities in the knowledge graph, and use these embeddings to learn a prediction function $\hat{y}_{uv} = f(u, v)$, where $\hat{y}_{uv}$ denotes the probability that user u will be interest by item v.

B. *GNN-based auto-encoder*

This section describes the GNN-based auto-encoder used to generate embeddings for the nodes in the KG. Compared to existing Auto-encoder-based structures, our model is designed to capture and utilize the rich information in knowledge graphs more effectively, thereby enhancing the performance of recommender systems.

Inspired by the Variational Graph Auto-Encoder (VGAE) [16], our model employs a variational mechanism in the encoder, which outputs a distribution in the embedding space for each node instead of a single point. This allows the model to capture the uncertainty in the node representations.

The encoder consists of three Graph Attention Networks (GAT) layers, where the first layer serves as a hidden layer, and the second and third layers output the mean and standard deviation of the distribution for each node, respectively.

GAT is a type of neural network architecture specifically designed for machine learning tasks on graph-structured data. Compared to another classic GNN architecture GCN, which uses edge weights to average neighborhood information, GATs are unique in their ability to assign varying degrees of importance, or "attention scores", to different nodes in a graph during the aggregation of neighborhood information.

This allows GATs to handle complex dependencies between nodes, providing a more nuanced understanding of the graph structure. The attention mechanism of GATs is data-driven, meaning it can adaptively focus on the most informative parts of the input graph, which can lead to better performance on various tasks. The aggregation process of a multi-head GAT layer is illustrated in Fig. 1.

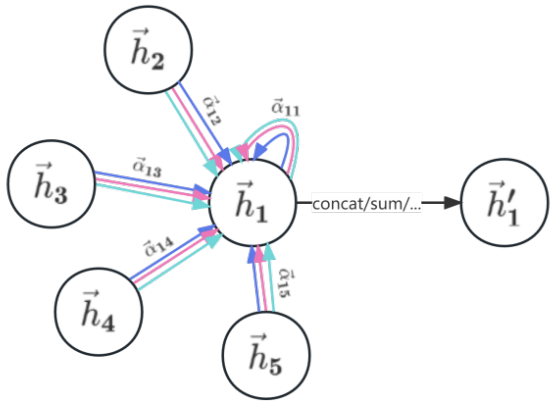

Figure 1 An illustration of multi-head attention (with 3 heads) by node 1 on its neighborhood. Different arrow colors denote independent attention computations.

The use of GATs allows the model to assign different weights to different neighbors when aggregating information, which effectively captures the heterogeneity of graph structures. The use of a variational mechanism enables the model to handle the uncertainty in the node representations, which is especially beneficial in the presence of noisy or incomplete information.

The decoder takes the embeddings of the head, relation, and tail of each triplet in the KG and generates a score indicating the likelihood of this triplet. The score is computed by first concatenating the embeddings of the head, relation, and tail, and then passing them through a two-layer Multi-Layer Perceptron (MLP) with a Gaussian Error Linear Unit (GELU) activation function in between. The final score is obtained by applying a sigmoid activation function to the output of the MLP. The GELU activation is defined as follow where *erf* is the Gauss error function.

$$GELU(x) = \frac{1}{2}x(1 + erf(\frac{x}{\sqrt{2}})) \quad (2)$$

GELU is a recently proposed activation function that has gained popularity in deep learning. It is a smooth approximation of the ReLU activation function. The GELU function has a non-monotonic, sigmoid-like shape that provides a better gradient flow during backpropagation compared to ReLU.

The Fig. 2 illustrate the whole process of the proposed GNN-based auto-encoder.

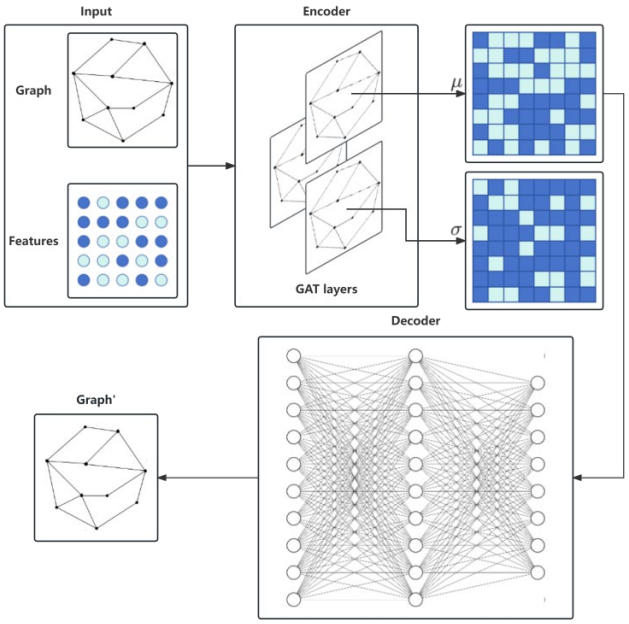

Figure 2 GNN-based auto-encoder

The auto-encoder is trained to reconstruct the input KG. In essence, it computes the plausibility of each triplet in the graph. Consequently, the loss function for this KGE model is formulated as a binary cross-entropy:

$$L_{KGE} = -\sum_{(e,r,e')\in G} \log(\hat{y}_{ere'}) - \sum_{(e,r,e')\notin G} \log(1-\hat{y}_{ere'}) \qquad (3)$$

where $\hat{y}_{ere'}$ is the predicted probability of triplet (e,r,e'). The first term quantifies the model's proficiency in identifying true edges in the KG, while the second term assesses its ability to discern false edges generated through negative sampling.

In addition to the KGE loss, the incorporation of the variational mechanism involves a Kullback-Leibler (KL) divergence loss. This loss quantifies the discrepancy between the learned distribution of the latent variables and the standard normal distribution.

$$L_{KL} = -\frac{1}{2N}\sum_{e\in G}(1+2\log\sigma_e - \mu_e^2 - \sigma_e^2) \qquad (4)$$

where N is the number of nodes in KG.

Compared to existing auto-encoder-based recommender systems, our approach uses Graph Neural Networks to aggregate network information, resulting in a more powerful knowledge graph embedding representation. In contrast to other graph representation auto-encoders such as VGAE, which can only handle simple undirected graphs and do not consider the types of edges, our encoder is more suited to the nature of knowledge graphs. When propagating information between nodes, our GNN-based auto-encoder uses GAT layers, which also incorporate edge features into the calculation. Furthermore, our decoder employs a MLP instead of a simple dot product, which is better suited to discerning relationships within complex graph structures.

C. *GraphRipple*

This section elaborates on how our GNN-based auto-encoder is incorporated with the user preference propagation mechanism. Drawing inspiration from RippleNet, this mechanism propagates a user's preference across the KG by simulating the spread of 'ripples'. The model operates on the assumption that a user's preferences can ripple out to distant items via the links in the KG.

Fig. 3 shows the overall structure of GraphRipple.

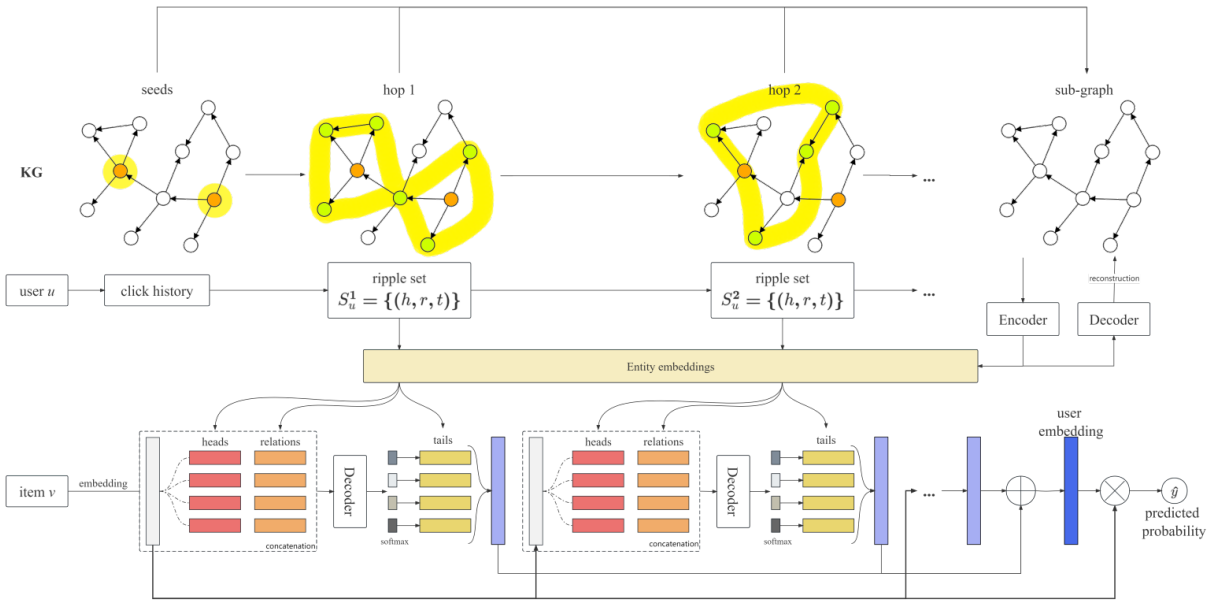

Figure 3 The overall structure of the GraphRipple

As depicted in Fig. 3, the propagation process continues for a predetermined number of layers. During this process, the user's preference distribution is gradually disseminated throughout the KG, thereby reaching an increasingly large number of items. Given the number of propagation layers, this process inherently forms a sub-graph of the KG. For every such sub-graph, we employ the auto-encoder model to learn the embeddings for all nodes and edges within it. Subsequently, we feed the embeddings of the head nodes and edges, concatenated with the item embedding in each instance, into the decoder. The relevance scores for each tail node are then obtained by applying the softmax function. The local user embedding of the current hop is thus determined by taking the weighted average over the tail embeddings.

Here, it is worth noting that an essential aspect to consider when working with GNNs is the size of the graph being processed. In many cases, the complete graph is too large to be effectively trained as a whole, much like how data needs to be divided into batches for traditional deep learning models. This is where the user preference propagation mechanism truly shines in its compatibility with GNNs. Preference propagation naturally generates sub-graphs of the KG through the spread of ripples. This sub-graph creation process aligns perfectly with the need for graph partitioning in GNN training. These sub-graphs are more manageable in size, making them ideal for processing with GNNs. Thus, this structure not only facilitates the propagation of user preferences but also inherently caters to the practical needs of GNN training. This makes it an ideal framework for integrating with GNNs, leading to a powerful model that leverages the strengths of both.

Returning to our main discussion, following the propagation process, GraphRipple constructs the user's preference vector by summing all local user embeddings. This vector serves as a comprehensive representation of the user's preference that has propagated through the KG.

To predict the probability of a user selecting an item, GraphRipple computes the dot product between the user's preference vector and the item's embedding vector, which is then passed through a sigmoid activation function. Consequently, items are recommended to the user based on these computed probabilities.

The loss function for the recommendation task is a binary cross-entropy:

$$L_{rec} = \sum_{(u,v)\in Y} -\left(y_{uv}\log\hat{y}_{uv} + (1-y_{uv})\log(1-\hat{y}_{uv})\right) \tag{5}$$

The complete loss function used for training GraphRipple is then as follow:

$$\begin{aligned} L &= L_{rec} + L_{KG} + L_{KL} + L_{reg} \\ &= \sum_{(u,v)\in Y} -\left(y_{uv}\log\hat{y}_{uv} + (1-y_{uv})\log(1-\hat{y}_{uv})\right) \\ &+\lambda_1\left(-\sum_{(e,r,e')\in G}\log(\hat{y}_{ere'}) - \sum_{(e,r,e')\notin G}\log(1-\hat{y}_{ere'})\right) \\ &+\lambda_2\left(-\frac{1}{2N}\sum_{e\in G}(1+2\log\sigma_e-\mu_e^2-\sigma_e^2)\right) \\ &+\frac{\lambda_3}{2}\|\Theta\|_2^2 \end{aligned} \tag{6}$$

where Θ represents all parameters of GraphRipple, and the forth term is the regularizer.

IV. EXPERIMENTS

A. Datasets

In order to evaluate the performance of our GraphRipple model, we conducted experiments on three different datasets: Book-Crossing, MovieLens-100k, and Last.FM, which represent the domains of books, movies, and music respectively.

Book-Crossing: This dataset contains 1,149,780 explicit book ratings from the Book-Crossing community .

MovieLens-100k: This is a well-known dataset for movie recommendations. It consists of 100,000 ratings from 943 users on 1682 movies [17].

Last.FM: This is a music dataset derived from the Last.fm music platform.

Table 1 Experimental Results

Model	Book-Crossing			MovieLens-100k			Last.FM		
	AUC	ACC	F1	AUC	ACC	F1	AUC	ACC	F1
SVD	0.672		0.635	-	-	-	0.769	-	0.696
LibFM	0.691	0.639	0.618	-	-	-	0.778	-	0.710
DKN	0.621	0.598	-	-	-	-	-	-	-
PER	0.617	0.588	0.562	-	-	-	0.633	-	0.596
CKE	0.674	0.635	0.611	-	-	-	0.744	-	0.673
GHRS	-	-	-	-	-	0.785	-	-	-
KGCN	0.722	-	0.682	-	-	-	0.774	-	0.692
Ripp-MKR	0.740	**0.712**	-	-	-	-	0.799	**0.756**	-
RippleNet	0.729	0.662	0.650	0.878	0.801	0.807	0.780	0.740	0.702
GraphRipple	**0.742**	0.695	**0.683**	**0.895**	**0.814**	**0.812**	**0.813**	0.753	**0.749**

B. Experiments Setup

The experiments are conducted within the context of click-through rate (CTR) prediction, implying that the models are trained to predict each interaction in the test set. For the purpose of evaluation, we employ the Area Under the Curve (AUC), Accuracy (ACC), and the F1 score as our metrics.

Tab. 1 provides the hyperparameter settings for each dataset. In this table, *d* represents the dimension for all embeddings, *d_h*

stands for the hidden dimension of the encoder, and *heads* refers to the number of heads in the multi-head attention layers. *H* is the number of hops and *k_h* is the node capacity for each hop. *d_r* signifies the dropout rate of the decoder's multi-layer perceptron (MLP), while η corresponds to the learning rate. The hyperparameters are chosen to get the best AUC.

Table 2 Hyperparameter Settings

Dataset	Hyperparameters
Book-crossing	d=64, d_h=32, *heads*=8, H=2, k_h=10, λ_1=0.02, λ_2=10^{-4}, λ_3=10^{-5}, d_r=0.45, η=2×10^{-4}
MovieLens-100k	d=32, d_h=16, *heads*=8, H=3, k_h=11, λ_1=0.05, λ_2=10^{-4}, λ_3=10^{-6}, d_r=0.2, η=0.01
Last.FM	d=64, d_h=64, *heads*=6, H=3, k_h=9, λ_1=0.01, λ_2=10^{-3}, λ_3=10^{-7}, d_r=0.15, η=10^{-3}

C. Results

Tab. 2 showcases the experimental results obtained from GraphRipple in comparison with various baseline models. We have chosen the following baseline models for our experiments: SVD[18], LibFM[19], DKN[20], PER[21], CKE[5], GHRS, KGCN[22], Ripp-MKR[23] and the original RippleNet[2].

The results for the baseline models are primarily sourced from [2, 22]. Due to variations in the datasets and metrics used in these references, certain entries in the table are left blank. To ensure a comprehensive comparison, we implemented a RippleNet model to obtain metrics not provided in the original paper.

Upon careful analysis, the experimental results clearly show that our GraphRipple model achieves superior performance on most metrics across different datasets when compared with all baseline models. Even though it doesn't outperform in every metric on every dataset, the overall performance of our model is highly competitive. This serves as solid evidence of the efficacy of our proposed method. A comparison between GraphRipple and the original RippleNet underscores the fact that a GNN-based KGE model can significantly enhance the performance of the recommendation task.

The superior performance of GraphRipple may be attributable to the enriched information captured in the embeddings generated by the GNN-based KGE model. This model, underpinned by a variational graph auto-encoder architecture with GAT layers, aggregates information from all nodes in the neighboring sub-graph to generate entity embeddings. This feature aligns seamlessly with the user preference propagation process, resulting in an optimized system. Consequently, the GraphRipple model can provide more precise and personalized recommendations, thus explaining its superior performance.

V. CONCLUSIONS

In conclusion, this study introduced GraphRipple, a novel recommendation model that leverages a GNN-based Knowledge Graph Embedding model and the RippleNet structure for efficient user preference propagation. Our experimental results, as compared to several baseline models, have convincingly demonstrated the superior performance of GraphRipple across multiple evaluation metrics.

The superior performance of GraphRipple can be attributed to its ability to generate rich and informative embeddings through the GNN-based KGE model. Moreover, the inherent compatibility between the user preference propagation process and the GNN-based KGE model results in an optimized system that significantly enhances the performance of the recommendation task.

Despite the promising results, there are several avenues for future work. One possible direction is to explore more sophisticated GNN architectures or propagation mechanisms that could further enhance the performance of the recommendation model. In addition, it would also be worthwhile to investigate how to incorporate other types of side information, such as user and item attributes, into the GraphRipple model to provide even more personalized recommendations.

Finally, while our current model has demonstrated its effectiveness on click-through rate prediction tasks, we are also interested in testing its performance on other types of recommendation tasks, such as rating prediction and top-N item recommendation. By doing so, we hope to further validate the versatility and robustness of the GraphRipple model.

REFERENCES

[1] Sun, Z., Guo, Q., Yang, J., Fang, H., Guo, G., and Zhang, J., 2019 Research commentary on recommendations with side information: a survey and research directions. Electronic Commerce Research and Applications, 37: 100879. https://doi.org/10.1016/j.elerap.2019.100879

[2] Wang, H., Zhang, F., Wang, J., Zhao, M., Li, W., and Xie, X., 2018. Ripplenet: propagating user preferences on the knowledge graph for recommender systems. In: Proceedings of the 27th ACM international conference on information and knowledge management. 417-426. https://dl.acm.org/doi/abs/10.1145/3269206.3271739

[3] Wang, H., Zhang, F., Zhang, M., Leskovec, J., Zhao, M., and Li, W., 2019. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery \& data mining. 968-977. https://doi.org/10.1145/3292500.3330836

[4] Alhijawi, B., and Kilani, Y., 2020 The recommender system: a survey. International Journal of Advanced Intelligence Paradigms, 15, (3): 229-251. https://doi.org/10.1504/IJAIP.2020.105815

[5] Zhang, F., Yuan, N.J., Lian, D., Xie, X., and Ma, W., 2016. Collaborative knowledge base embedding for recommender systems. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 353-362. https://doi.org/10.1145/2939672.2939673

[6] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O., 2013 Translating embeddings for modeling multi-relational data. Advances in Neural Information Processing Systems, 26. https://dl.acm.org/doi/abs/10.5555/2999792.2999923

[7] Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X., 2015. Learning entity and relation embeddings for knowledge graph completion. In: Proceedings of the AAAI conference on artificial intelligence. https://doi.org/10.1609/aaai.v29i1.9491

[8] Sedhain, S., Menon, A.K., Sanner, S., and Xie, L., 2015. Autorec: autoencoders meet collaborative filtering. In: Proceedings of the 24th international conference on World Wide Web. 111-112. https://doi.org/10.1145/2740908.2742726

[9] Darban, Z.Z., and Valipour, M.H., 2022 Ghrs: graph-based hybrid recommendation system with application to movie recommendation. EXPERT SYSTEMS WITH APPLICATIONS, 200: 116850. https://doi.org/10.1016/j.eswa.2022.116850

[10] Kipf, T.N., and Welling, M., 2016 Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907. https://doi.org/10.48550/arXiv.1609.02907

[11] Hamilton, W., Ying, Z., and Leskovec, J., 2017 Inductive representation learning on large graphs. Advances in Neural Information Processing Systems, 30. https://doi.org/10.48550/arXiv.1706.02216

[12] Veli V C Kovi C, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y., 2017 Graph attention networks. arXiv preprint arXiv:1710.10903. https://doi.org/10.48550/arXiv.1710.10903

[13] Battaglia, P., Pascanu, R., Lai, M., Jimenez Rezende, D., and Others, 2016 Interaction networks for learning about objects, relations and physics. Advances in Neural Information Processing Systems, 29. https://dl.acm.org/doi/abs/10.5555/3157382.3157601

[14] Fout, A., Byrd, J., Shariat, B., and Ben-Hur, A., 2017 Protein interface prediction using graph convolutional networks. Advances in Neural Information Processing Systems, 30. https://dl.acm.org/doi/10.5555/3295222.3295399

[15] Hamaguchi, T., Oiwa, H., Shimbo, M., and Matsumoto, Y., 2017 Knowledge transfer for out-of-knowledge-base entities: a graph neural network approach. arXiv preprint arXiv:1706.05674. https://doi.org/10.48550/arXiv.1706.05674

[16] Kipf, T.N., and Welling, M., 2016 Variational graph auto-encoders. arXiv preprint arXiv:1611.07308. https://doi.org/10.48550/arXiv.1611.07308

[17] Harper, F.M., and Konstan, J.A., 2015 The movielens datasets: history and context. Acm transactions on interactive intelligent systems (tiis), 5, (4): 1-19. https://doi.org/10.1145/2827872

[18] Koren, Y., 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. 426-434. https://doi.org/10.1145/1401890.1401944

[19] Rendle, S., 2012 Factorization machines with libfm. ACM Transactions on Intelligent Systems and Technology (TIST), 3, (3): 1-22. https://doi.org/10.1145/2168752.2168771

[20] Wang, H., Zhang, F., Xie, X., and Guo, M., 2018. Dkn: deep knowledge-aware network for news recommendation. In: Proceedings of the 2018 world wide web conference. 1835-1844. https://doi.org/10.1145/3178876.3186175

[21] Yu, X., Ren, X., Sun, Y., Gu, Q., Sturt, B., and Khandelwal, U., 2014. Personalized entity recommendation: a heterogeneous information network approach. In: Proceedings of the 7th ACM international conference on Web search and data mining. 283-292. https://doi.org/10.1145/2556195.2556259

[22] Wang, H., Zhao, M., Xie, X., Li, W., and Guo, M., 2019. Knowledge graph convolutional networks for recommender systems. In: The world wide web conference. 3307-3313. https://doi.org/10.1145/3308558.3313417

[23] Wang, Y., Dong, L., Li, Y., and Zhang, H., 2021 Multitask feature learning approach for knowledge graph enhanced recommendations with ripplenet. PLoS One, 16, (5): e251162. https://doi.org/10.1371/journal.pone.0251162