RKGREC: A Rule-guided Graph Neural Network Recommendation Model

Elara Everly (ElaraEverly92gsu@hotmail.com)
Georgia State University
Arwen Sinclair
Georgia State University
Lachlan Beckett
Georgia State University
Sullivan Kingsley
Georgia State University

Research Article

Keywords: Rule-guided, Recommendation, Graph neural networks, Learning, Feature

Posted Date: June 27th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3108952/v1

RKGREC: A Rule-guided Graph Neural Network Recommendation Model

Elara Everly, Arwen Sinclair, Lachlan Beckett,Sullivan Kingsley,

Abstract—With the proliferation of the Internet and the vast amount of information available, recommendation systems have become indispensable for filtering and delivering personalized content to users. Collaborative filtering, one of the most widely used recommendation algorithms, predicts user preferences based on historical interactions. However, as the number of users and items increases, challenges such as sparsity and cold start arise, hindering personalized recommendations. To address these challenges, recommendation algorithms have started incorporating knowledge graphs, structured data repositories, into their models. These algorithms can be classified into embedding-based, rule-based, and aggregation-based approaches. Embedding-based algorithms leverage existing knowledge graph embedding models to obtain entity embeddings, which are then used as input features for recommendation prediction. Rule-based algorithms focus on modeling connections between users and items using paths in the knowledge graph. They can be categorized as meta-path-based or automatic path mining approaches. Aggregation-based algorithms aggregate entities using graph neural networks, such as graph convolutional neural networks (GCNs) or graph attention networks (GATs), to capture relationships between entities at different distances. However, while aggregation-based approaches excel in modeling accuracy, they may lack generalization capability. To overcome this limitation, this paper introduces RKGREC, a recommendation algorithm that combines rule-based modeling with graph neural networks. By leveraging the strengths of both approaches, RKGREC achieves high modeling accuracy through rules while maintaining high generalization capability through entity aggregation. The integration of rules and graph neural networks enhances the semantic understanding of connections between users and items, reducing noise and improving recommendation performance.

Index Terms—Rule-guided, Recommendation, Graph neural networks, Learning, Feature.

1 Introduction

With the exponential growth of the Internet, individuals now have access to an overwhelming amount of information encompassing various media types such as news, music, videos, and products. Consequently, extracting the desired content from this vast pool of web data has become a daunting task, giving rise to the development of recommendation systems. The ubiquity of recommendation systems is evident in popular online platforms like YouTube and Netflix, where users are presented with personalized video recommendations.

In today's internet landscape, recommendation systems have assumed a pivotal role across numerous domains, ranging from information sharing platforms [1] and e-commerce to computational advertising [2]. They have become an indispensable component of the information era. Essentially, a recommendation system employs algorithms to anticipate a user's preferences or interests, essentially serving as an intelligent information filtering system. For instance, on YouTube, a recommendation system's primary objective is to suggest videos that align with a user's preferences. In the realm of computational advertising, recommendation systems aim to discern users' interests and deliver tailored advertisements accordingly. Collaborative filtering [3] is a highly popular recommendation algorithm extensively utilized in recommendation systems. The fundamental concept revolves around discovering correlations among users based on their item preferences, thereby making recommendations based on these correlations. In simpler terms, users with similar preferences tend to exhibit similar behavior. Collaborative filtering primarily relies on historical interaction data between users and items.

Traditionally, collaborative filtering models are built upon the historical interaction information between users and items, such as ratings or click-through rates. These models derive embedding representations for users and items. For instance, matrix decomposition [4] maps users and items into a shared latent space, enabling the calculation of user preferences for items using inner product operations. Recent advancements in deep learning have further enhanced the performance of recommendation systems. Notably, the NCF model proposed by He et al. [5] combines traditional matrix decomposition with multilayer perceptrons (MLPs) to optimize the conventional approach. Similarly, the NGCF model proposed by Wang et al. [6] constructs a bipartite graph of user-item interactions and embeds these interactions into feature vectors using graph models. While these methods have improved the performance of recommendation systems, challenges arise as the number of users and items increases. The system may encounter issues of sparsity and cold start, leading to difficulties in achieving personalized recommendations. The cold start problem occurs when the number of users and their interactions with items is limited during the initial operation of the recommender system [7]. Collaborative filtering algorithms struggle in cold start scenarios where

[*]Elara Everly is the corresponding author.

• *Elara Everly, Arwen Sinclair, Lachlan Beckett, and Sullivan Kingsley are with the Georgia State University. (e-mail: ElaraEverly92gsu@hotmail.com).*

user-item interaction history is sparse, making it challenging to provide recommendations for new users or new items. To mitigate the cold start problem, many algorithms incorporate external resources, such as users' and items' attributes, to reduce the reliance on collaborative filtering solely based on historical behavior. This approach leverages knowledge and common sense to infer user interests. Various external resources can be utilized, including text, video, and images. Among these resources, structured data like knowledge graphs are often favored due to their extensive coverage and high data quality [8].

Recommendation algorithms that leverage knowledge graphs can be categorized into three main types: embedding-based algorithms, rule-based algorithms, and aggregation-based algorithms. These algorithms typically begin by mapping items in the recommendation dataset to corresponding entities in the knowledge graph. Embedding-based algorithms utilize knowledge graph embedding models, such as TransE [2], to capture the structural relationships within the knowledge graph. The embeddings of entities associated with items in the knowledge graph are then used as input features for recommendation prediction. This can be achieved by incorporating existing mature algorithms [9] or designing new algorithms [10] specifically for this purpose. Rule-based algorithms focus on modeling the connections between users and items by considering the paths between users and entities as features. These approaches can be further categorized into meta-path based approaches [11, 12, 13, 14], which rely on predefined meta-paths, and automatic path mining approaches [15, 16] that automatically discover relevant paths. Aggregation-based methods place emphasis on aggregating entities. Representative approaches include RippleNet [17], KGCN [18], and KGAT [19]. These methods employ self-designed aggregation models, such as graph convolutional neural networks (GCNs) [20] or graph attention networks (GAT) [21], to iteratively aggregate neighboring entities around a central entity at different distances. This enables the central entity to gather information from its surrounding nodes. While aggregation-based modeling approaches achieve high modeling accuracy, they may lack generalization capability. Although these models effectively capture connections between multiple entities, their aggregation methods tend to randomly mix information and lack the selectivity of rule-based approaches. Additionally, they may not have a clear understanding of the actual connections between distant nodes and central nodes, potentially introducing noise. Consequently, these modeling approaches exhibit strong generalization abilities but may lack accuracy. To address the limitations of existing recommendation algorithms, this paper proposes a novel recommendation algorithm called RKGREC. RKGREC combines the strengths of rules and graph neural networks to enhance its modeling capabilities. Rules provide RKGREC with accurate linear modeling capabilities, while graph neural networks offer high generalization capabilities through aggregation modeling. Rules not only capture long-range semantics between users and entities but also assist graph neural networks in selectively aggregating entities, establishing clear semantic links between distant and central nodes, and reducing the introduction of noise.

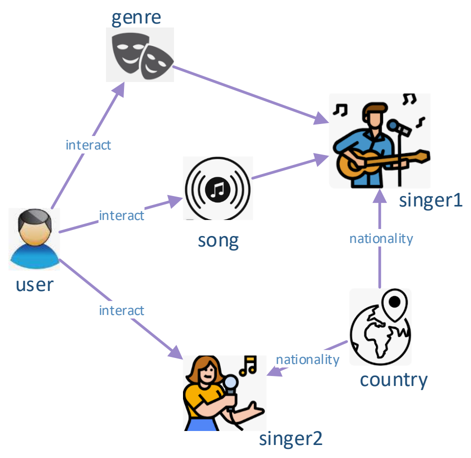

Fig. 1: Illustration of Knowledge Graph.

2 Related Work

2.1 Collaborative filtering recommendation algorithm

The collaborative filtering recommendation algorithm is a well-established method in the field of recommendation systems. It can be broadly categorized into two main types: domain-based collaborative filtering algorithms and model-based collaborative filtering algorithms. The model-based approach encompasses various techniques such as graph models, clustering models [22], Bayesian models [23], and matrix decomposition models [4]. These models aim to learn a predictive function that effectively captures the interaction patterns between users and items. By utilizing this function, the algorithm can predict scores for missing items in the recommendation matrix.

In recent times, many recommendation algorithms have sought to enhance the accuracy of their predictions by incorporating additional information about users and items. For instance, Ling et al. [24] proposed a model that combines the collaborative filtering algorithm with review text information, taking into account both the textual content and the rating information. Ma et al. [25] introduced a model that integrates social factors into the matrix decomposition algorithm, considering the influence of social connections and other factors. Ye et al. [26] developed a POI (Point of Interest) recommendation model that incorporates user preferences and social factors. Furthermore, Rendle et al. [27] emphasized the importance of feature engineering and proposed a factor decomposer algorithm.

Overall, collaborative filtering remains a foundational technique in recommendation systems [28], and researchers continue to explore ways to leverage additional information to enhance its accuracy and effectiveness.

2.2 Knowledge Graph-based Recommendation

Embedding-based algorithms draw on the idea of knowledge graph embedding tasks. Such algorithms directly use knowledge graph embedding models such as TransE [2] and TransR [29] to generate knowledge graph embeddings, and design algorithms to improve the prediction accuracy of the recommendation system by using the knowledge graph embedding as an input feature to provide additional knowledge of items to the recommendation system. For example, MKR designed a separate cross & compress unit to make the recommendation task better by sharing the feature parameters of both the recommendation and knowledge graph embedding tasks, so that the recommendation task gets the help of the knowledge graph embedding task. CKE also inputs the visual knowledge related to items, textual knowledge, and structured knowledge from the knowledge graph as features into the collaborative filtering algorithm to improve the predictive power of the collaborative filtering algorithm. Among them, CKE uses TransR to generate embeddings for structured knowledge in the knowledge graph. DKN [30] uses TransD [31] to generate embeddings of entities in the news and uses embeddings generated from sentences, characters and entities in the news to construct news KSR [32] uses TransE to generate entity embeddings and designed the KeyValue Memory Network to exploit the knowledge in the knowledge graph. KTUP [33] uses TransH [34] to learn both recommendation models and knowledge graph completion. Knowledge Graph Completion (KGC) model [35].

In recommender systems, a meta-path is usually defined as a sequence of entity types between a user and an item; for example, $user \rightarrow song \rightarrow singer \rightarrow song$ can be used as a meta-path between a user and a song. PRE [11] uses meta-path-based similarity to calculate the recommendation probability of an item. FMG constructs a meta-path by combining multiple metagraphs and further by using meta path-based similarity to make recommendations. HERec [?] and Metapath2vec [13] use meta paths for random wandering and perform embedding generation of entities from the points sampled by the random wandering. Metapaths are constructed by experts, so these meta paths tend to be ideographically clear and have a reasonable length structure, which is the biggest advantage of methods that use meta paths. The biggest disadvantage of such methods is their reliance on human resources. The performance of meta path-based methods often depends heavily on the quality of the constructed meta paths, requiring considerable human resources to test and modify the meta paths, and requiring experts to construct a new set of meta paths for almost every new dataset. To address the shortcomings of meta-path-based approaches, RKGE and KRPN automatically my paths under certain path length constraints and use recurrent neural networks to model the path information. These algorithms improve the heavy reliance on human labor of meta-path-based algorithms to some extent, however, they have some shortcomings in the screening of mined paths. For all paths between users and items, RKGE samples only a few, e.g., 5 if the maximum length of the rule is 3. The features of the sampled paths are also not utilized in their entirety; instead, multiple feature vectors are compared in each dimension using a maximum pooling layer, and only the most significant features are retained. KRPN considers the entire set of rules and uses a gradient update strategy with weights to differentiate the importance, which can be interpreted as assigning different weights to different paths.

RippleNet [17] classifies the entities around an entity into 1hop neighbor nodes, 2hop neighbor nodes, . . . , and k hop neighbor nodes, and then aggregates the nodes located at different hop with weights to obtain the feature representation of the entity. Meanwhile, a user's embedding is formed by combining the embeddings of multiple entities clicked by that user. KGCN [34] and KGCNLS use graph convolutional neural networks to aggregate the nodes around an entity to that entity. These two works use an attention mechanism to construct relationships between entities as weights in entity aggregation so that explicit long-range semantics between users, and entities are ignored and it is difficult to clarify what semantic connections are made between points that are distant from the central node and the central node. Meanwhile, KGCN and KGCNLS sample the points for aggregation randomly, which may cause information loss. KGAT [19] uses GAT as the base model for node aggregation, and GAT takes the whole knowledge graph as input, and instead of sampling points for aggregation, all the points around a node are aggregated to that point, which increases the computational burden on one hand, and on the other hand, aggregating all the points introduces a lot of noise because not On the other hand, aggregating all the points introduces a lot of noise because not all the points are loaded with sufficient amount of information.

3 METHODOLOGY

3.1 Construction of knowledge graph

It is evident that a user's preference for a product can be inferred from the vector dot product between the user and the product. However, traditional recommendation algorithms, like matrix decomposition, do not embed the interaction information directly into the feature vector. Instead, they initialize the vector with attributes such as ID and subsequently optimize the model using the interaction information.

3.2 Sampling and grouping neighbors

In order to generate low-dimensional vector representations (e_v) for nodes in a graph, a network representation learning approach is employed. This technique aims to efficiently capture the structural properties and semantic information of heterogeneous graphs. The resulting node embeddings can then be effectively utilized in recommendation models. To obtain the node embeddings, a restart-based random wander sampling strategy is employed, which involves iteratively traversing the graph starting from a randomly chosen node, denoted as the starting node (v). During the random wander, with a probability P, the traversal proceeds to one of the neighboring nodes of the current node, or it returns to the starting node. This process is repeated until a fixed number of nodes, denoted as R, is successfully sampled. To ensure a balanced representation, the number of nodes from different types in each iteration is limited, thereby guaranteeing that nodes of all types are sampled. After sampling, the neighbor nodes are classified according to their types. The top-k nodes, determined

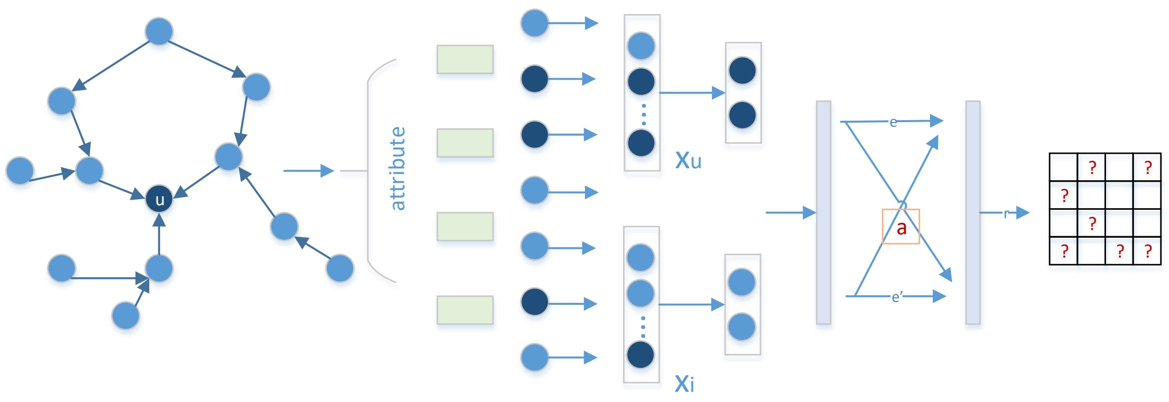

Fig. 2: Schematic diagram of our model.

based on their frequency of occurrence in R, are selected as the set of neighbors associated with the node V of type t. These selected neighbors provide valuable auxiliary information for the recommendation model. By utilizing network representation learning methods, the generated node embeddings capture both the structural properties and semantic information of the heterogeneous graph. This approach offers an improvement over traditional representation learning methods, enabling the node embeddings to be effectively applied in recommendation models.

3.3 Embedding Layer

The embedding layer transforms the input sparse vectors into dense vectors and randomly initializes the vector matrices of users and items $E \in R^{(M+N)\times d}$. Where the number of users is M, the number of items is N, and d is the dimensionality of the representation size.

$$E = [e_{u1}, \ldots, e_{uM}, e_{i1}, \ldots, e_{iN}] \tag{1}$$

3.4 Propogation Layer

Given the adjacency matrix representing the connections between users and products, the feature vectors of nodes are fused with the feature vectors of their neighboring nodes. This fusion process aims to integrate the information from adjacent nodes and enhance the representation of each node. Subsequently, the fused feature vectors are passed through a multilayer neural network to obtain the final feature vectors. In the context of user-item pairs, for a specific node pair (u, i), where u represents the user node and i represents the item node, a message transfer function is defined to capture the information flow from item i to user u. This function facilitates the propagation of relevant information from the item node to the user node in the graph. The purpose of this message transfer function is to enable the user node to receive valuable information and characteristics from the item node, thereby enhancing the representation of the user's preferences or interactions with the item. The function leverages the adjacency matrix and other relevant features to facilitate the information transfer process. By applying this message transfer function to all user-item pairs in the graph, the fusion and information propagation processes collectively contribute to the generation of informative and comprehensive feature vectors that encapsulate the characteristics and relationships between users and items.

$$m_{u,i]} = \frac{1}{\sqrt{|N_i||N_u|}}(W_1 e_i + +W_2(e_i \odot e_u)) \tag{2}$$

where is the feature vector of e_i items, e_u is the feature vector of users. $\frac{1}{\sqrt{|N_i||N_u|}}$ is the weight coefficient between user u and commodity i. N_u and N_i represent the number of neighbor nodes of user and commodity respectively, and $W_1, W_2 \in R^d$ is the weight matrix parameter. $\odot$ is the product of the corresponding elements of the vectors.

In the case where a user node has multiple neighboring nodes, the final vector representation of the user is obtained by fusing the feature vectors of all its neighbor nodes. This fusion process ensures that the user's representation incorporates the information from its immediate neighbors. By utilizing layer-wise information transfer, the user node is able to acquire information from its higher-order neighbors in the graph. This enables the user to gain a broader understanding of its surrounding nodes and their characteristics. The feature vector representation of the user at the first layer, as depicted in Equation 3, demonstrates this information transfer process. It is important to note that during the information transfer process, the user's own features are retained and combined with the features of its neighboring nodes. This ensures that the final vector representation of the user not only incorporates the information from its neighbors but also retains the user's own characteristics. By integrating both the user's features and the features of its neighbors, the user's representation becomes more comprehensive and informative.

$$e_u^{(l)} = LeakyReLU(m_{u,u}^{(l)} + \sum_{i \in N_u} m_{u,i}^{(l)}) \tag{3}$$

$$m_{u,i}^{(l)} = p_{ui}(W_1^{(l)} e_i^{(l-1)} + W_2^{(l)}(e_i^{(l-1)} \odot e_u^{(l-1)})) \tag{4}$$

$$m_{u,u}^{(l)} = W_1^{(l)} e_u^{(l-1)} \tag{5}$$

where p_{ui} is the weight coefficient. W_1, W_2 and are the matrix parameters. $e_i^{(l-1)}$ is the eigenvector of the items in the l-1th layer. $e_u^{(l-1)}$ is the eigenvector of the user in layer l-1.

After layer propagation, the user vector representation can be obtained as a set below:

$$E = [e_u^{(0)}, \dots, e_u^{(l)}] \tag{6}$$

The item vector representation can be obtained as a set below:

$$E = [e_i^{(0)}, \dots, e_i^{(l)}] \tag{7}$$

The final representation of the user and product feature vectors is shown below. Where || denotes the splicing between vectors.

$$e_u = e_u^{(l)} || e_u^{(l)} \tag{8}$$

$$e_i = e_i^{(l)} || e_i^{(l)} \tag{9}$$

3.5 Extraction and transformation of features

To capture the attribute information of neighboring nodes in graphs, the process of neighborhood aggregation is employed, specifically considering meta-paths. Meta-paths represent a sequence of node types and edges that define a semantic relationship between nodes. By aggregating the attribute information of neighbors along each meta-path for a target node, a comprehensive representation of the node can be obtained. The neighborhood aggregation process calculates the average of the attribute information of all paths in each meta-path connecting the user node v with its neighbors. This results in a comprehensive representation of the user node that incorporates the information from its neighbors based on the defined meta-paths. By considering the attribute information of neighboring nodes through meta-paths and performing neighborhood aggregation, a more holistic and informative representation of the target node can be obtained.

$$x_u^{\rho} = \sum_{i \in N_u^{\rho}} W_u^{\rho} \cdot x_u \tag{10}$$

where N_u^{ρ} denotes the neighbor aggregation of node u based on meta-path ρ and x_u denotes the attribute information vector of node v.

Feature transformation: for each node u, we can obtain the node's own representation vector x_u. We can also obtain the set of neighbor aggregation features of the node based on meta-paths. where ρ denotes meta-paths. In order to better represent the node effectively, it is necessary to transform and fuse the representation of the original attribute information.

$$x_u = M_{\rho} \cdot x_u^{\rho} \tag{11}$$

where x_u^{ρ} and x_u, are the original and transformed features of node v, respectively. With a specific type of projection shadowing operations, the hierarchical attention mechanism can handle arbitrary types of nodes.

3.6 Learning of Top-k Ranking Model

The user's final node embedding is used as the input to a multi-layer fully connected layer with a non-linear transformation. The form is shown below:

$$Z_u = ReLU(W_L + \cdots + ReLU(W_1 e_u + b_1) + b_L) \tag{12}$$

where W and b denote the weight matrix and bias vector of each layer, respectively.

$$p_u = \frac{1}{1 + exp(-(w_p^T Z_u + b_p))} \tag{13}$$

where w_p and b_p denote the weight matrix and bias vector, respectively, and great likelihood estimation is used to model the optimization objective of the recommendation task.

$$L = \sum_{u, y_u \in D} (y_u \log(p_u) + (1 - y_u) \log(1 - p_u) + \lambda ||\theta||^2) \tag{14}$$

where λ and θ are the set of parameters and regularization coefficients of the model, respectively. The parameters can be updated by the random gradient descent method.

4 EXPERIMENTS

4.1 Datasets

The comparison experiments involved three types of datasets: Movielens, LastFM, and Yelp. The statistical information of these datasets is summarized in Table 1. These statistics provide an overview of the datasets used in the comparison experiments, including the number of users, items, and interactions between them.

4.2 Metrics

To evaluate the performance of the recommendation model, the three datasets (Movielens, LastFM, and Yelp) were randomly divided into training and test sets. The division followed a ratio of 80% for the training set and 20% for the test set. For model training, 80% of the data from each dataset was used to train the recommendation model. This training data was utilized to optimize the model parameters and learn the underlying patterns and relationships within the datasets. After training the model, the remaining 20% of the data from each dataset was used as the test set. This test set served as an independent evaluation dataset to assess the performance and generalization capability of the trained model.

To evaluate the recommendation model's performance, metrics commonly used in ranking recommendation tasks were employed. Specifically, recall and normalized discounted cumulative gain (NDCG) were utilized as evaluation metrics. Recall measures the proportion of relevant items that are successfully recommended to users. It quantifies the ability of the model to capture and recommend items that the users would find relevant. NDCG takes into account both the relevance of the recommended items and their position in the recommendation list. It provides a more comprehensive evaluation by considering the graded relevance of items and penalizing lower-ranked items. By employing recall and NDCG as evaluation metrics, the performance of the recommendation model can

TABLE 1: Statistical information of datasets.

dataset	users	items	u-i interactions	u-u interactions	i-i interactions
Movielens	943	1682	100000	47150	82795
LastFM	1892	17632	92834	18802	153399
Yelp	16239	14284	198937	158590	14267

TABLE 2: Model Performance Comparison on Movielens.

Model	Recall	NDCG
ItemKNN	0.183	0.572
BPR	0.195	0.585
NeuMF	0.217	0.594
LRML	0.221	0.608
Ours	0.233	0.627

TABLE 3: Model Performance Comparison on LastFM.

Model	Recall	NDCG
ItemKNN	0.411	0.537
BPR	0.425	0.542
NeuMF	0.487	0.596
LRML	0.503	0.619
Ours	0.524	0.633

be assessed in terms of its ability to accurately recommend relevant items to users.

$$Recall = \frac{\sum_{u \in U} Rec_u \cap Test_u}{|\sum_{u \in U} Test_u|}, \tag{15}$$

$$NDCG = Z_k \sum_{k=1}^{K} \frac{2r_k - 1}{log_2(k+1)}, \tag{16}$$

Where "Rec" represents the list of recommendations provided to the user, while "Test" denotes the list of items clicked by the user. In this evaluation, the value of $r_k = 1$ indicates the user's interest in the k-th item from the recommendation list, whereas $r_k = 0$ signifies the user's lack of interest in the k-th item. Additionally, Z_k represents the normalization constant. The recall metric measures the coverage of the recommendation algorithm's results in meeting the user's needs, with higher recall values indicating a broader coverage of relevant recommendations.

4.3 Baselines

We compare our model with some relevant or current state-of-the-art baselines.

- ItemKNN: A classical collaborative filtering recommendation algorithm, based on the historical interaction behavior of users and items to make recommendations.
- BPR [27]: Bayesian-based personalized ranking model.
- NeuMF: Neural network based ranking recommendation algorithm with matrix decomposition and multilayer perceptron.
- LRML: Recommendation algorithm based on attention mechanism, mainly applied to Top-K ranking recommendation of items.

TABLE 4: Model Performance Comparison on Yelp.

Model	Recall	NDCG
ItemKNN	0.212	0.620
BPR	0.226	0.643
NeuMF	0.231	0.657
LRML	0.239	0.669
Ours	0.248	0.685

4.4 Experiment result analysis

The superior performance of the NeuMF model in the comparison approach can be attributed to several key factors. One of the main reasons is that the model incorporates a multi-layer perceptron (MLP) to model the interaction between users and items. This MLP-based modeling approach enhances the effectiveness of the model in capturing and representing the complex relationships and preferences between users and items. Our proposed model outperforms the other comparison methods across all three datasets, indicating its effectiveness in recommendation tasks. This superiority can be attributed to its ability to effectively leverage node neighborhoods and meta-path generation for feature representation. Firstly, the RKGREC model recognizes that different meta-paths have varying contributions to the representation of nodes (users), and these contributions are not treated equally. The model captures the varying importance and relevance of different meta-paths in characterizing the nodes, allowing for a more nuanced and comprehensive understanding of user preferences. Secondly, the model acknowledges that each node (user) along each meta-path may contribute different attributes or features of varying importance. This awareness of the diverse contributions of node attributes allows the RKGREC model to capture and utilize the most relevant and informative features for recommendation purposes. By considering the distinct contributions of different rule guided and the varying importance of node attributes, the proposed RKGREC model demonstrates its effectiveness in generating accurate and personalized recommendations. Its ability to leverage these two aspects of feature representation enhances its performance and sets it apart from the other comparison methods.

5 Conclusion and Future Work

Collaborative filtering, a widely used recommendation algorithm, leverages the correlation between users' item preferences to make recommendations. It relies on historical interaction data between users and items, such as ratings or click-through rates. Typically, collaborative filtering involves obtaining embedding representations of users and items. Matrix decomposition is an example, mapping users and items to a hidden space and calculating user preferences using inner product. Deep learning advancements have further enhanced recommendation systems, and algorithms utilizing knowledge graphs can be categorized into embedding-based, rule-based, and aggregation-based approaches. In embedding-based algorithms, the knowledge graph structure is modeled using techniques like TransE, obtaining embeddings for corresponding entities. These embeddings serve as input features for recommendation prediction, either through existing algorithms or novel designs. Rule-based algorithms focus on modeling user-item connections by considering paths between users and entities as features. While they achieve high modeling accuracy, their generalization capability is limited. On the other hand, aggregation-based models capture connections among multiple entities, but their approach tends to introduce noise as it randomly aggregates points. Although these models excel in generalization, they lack accuracy. To address these limitations, this paper introduces RKGREC, a recommendation algorithm that combines rules and graph neural networks (GNNs). By incorporating rules, RKGREC achieves accurate linear modeling, especially for long-range semantics between users and entities. GNNs complement the model with their ability to perform aggregation modeling, allowing high generalization capability. Rules also assist in selective entity selection during aggregation, establishing clear semantic links between distant and central nodes and reducing noise. The integration of rules and GNNs in RKGREC addresses the shortcomings of existing recommendation algorithms. It provides accurate linear modeling through rules while benefiting from the aggregation modeling capability and generalization of GNNs. This combination improves the overall performance and effectiveness of the recommendation system.

6 Conflict of interest statement

All authors have no conflict and declare that: (i) no support, financial or otherwise, has been received from any organization that may have an interest in the submitted work ; and (ii) there are no other relationships or activities that could appear to have influenced the submitted work.

References

[1] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM conference on recommender systems*, 2016, pp. 191–198.

[2] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, 2013.

[3] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, 2009.

[4] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[5] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[6] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 2019, pp. 165–174.

[7] Y. Wei, X. Wang, Q. Li, L. Nie, Y. Li, X. Li, and T.-S. Chua, "Contrastive learning for cold-start recommendation," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 5382–5390.

[8] Y. Wei, X. Wang, L. Nie, S. Li, D. Wang, and T.-S. Chua, "Causal inference for knowledge graph based recommendation," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[9] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 353–362.

[10] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, and M. Guo, "Multi-task feature learning for knowledge graph enhanced recommendation," in *The world wide web conference*, 2019, pp. 2000–2010.

[11] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han, "Personalized entity recommendation: A heterogeneous information network approach," in *Proceedings of the 7th ACM international conference on Web search and data mining*, 2014, pp. 283–292.

[12] C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip, "Heterogeneous information network embedding for recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 2, pp. 357–370, 2018.

[13] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 135–144.

[14] X. Yu, T. Gan, Z. Cheng, and L. Nie, "Personalized item recommendation for second-hand trading platform," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 3478–3486.

[15] X. Wang, D. Wang, C. Xu, X. He, Y. Cao, and T.-S. Chua, "Explainable reasoning over knowledge graphs for recommendation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 5329–5336.

[16] Z. Sun, J. Yang, J. Zhang, A. Bozzon, L.-K. Huang, and C. Xu, "Recurrent knowledge graph embedding for effective recommendation," in *Proceedings of the 12th ACM conference on recommender systems*, 2018, pp. 297–305.

[17] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo, "Ripplenet: Propagating user preferences on the knowledge graph for recommender systems," in *Proceedings of the 27th ACM international conference on information and knowledge management*, 2018, pp. 417–426.

[18] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo, "Knowledge graph convolutional networks for recommender systems," in *The world wide web conference*, 2019, pp. 3307–3313.

[19] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "Kgat: Knowledge graph attention network for recommendation," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 950–958.

[20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[21] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[22] S. Gong, "A collaborative filtering recommendation algorithm based on user clustering and item clustering." *J. Softw.*, vol. 5, no. 7, pp. 745–752, 2010.

[23] P. Zigoris and Y. Zhang, "Bayesian adaptive user profiling with explicit & implicit feedback," in *Proceedings of the 15th ACM international conference on Information and knowledge management*, 2006, pp. 397–404.

[24] G. Ling, M. R. Lyu, and I. King, "Ratings meet reviews, a combined approach to recommend," in *Proceedings of the 8th ACM Conference on Recommender systems*, 2014, pp. 105–112.

[25] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, "Recommender systems with social regularization," in *Proceedings of the fourth ACM international conference on Web search and data mining*, 2011, pp. 287–296.

[26] M. Ye, P. Yin, W.-C. Lee, and D.-L. Lee, "Exploiting geographical influence for collaborative point-of-interest recommendation," in *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, 2011, pp. 325–334.

[27] S. Rendle, "Factorization machines," in *2010 IEEE International conference on data mining*. IEEE, 2010, pp. 995–1000.

[28] Y. Wei, X. Wang, L. Nie, X. He, and T.-S. Chua, "Graph-refined convolutional network for multimedia recommendation with implicit feedback," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 3541–3549.

[29] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in

Proceedings of the AAAI conference on artificial intelligence, vol. 29, no. 1, 2015.

[30] H. Wang, F. Zhang, X. Xie, and M. Guo, “Dkn: Deep knowledge-aware network for news recommendation,” in *Proceedings of the 2018 world wide web conference*, 2018, pp. 1835–1844.

[31] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, “Knowledge graph embedding via dynamic mapping matrix,” in *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, 2015, pp. 687–696.

[32] J. Huang, W. X. Zhao, H. Dou, J.-R. Wen, and E. Y. Chang, “Improving sequential recommendation with knowledge-enhanced memory networks,” in *The 41st international ACM SIGIR conference on research & development in information retrieval*, 2018, pp. 505–514.

[33] Y. Cao, X. Wang, X. He, Z. Hu, and T.-S. Chua, “Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences,” in *The world wide web conference*, 2019, pp. 151–161.

[34] Z. Wang, J. Zhang, J. Feng, and Z. Chen, “Knowledge graph embedding by translating on hyperplanes,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 28, no. 1, 2014.

[35] Y. Wei, X. Wang, L. Nie, X. He, R. Hong, and T.-S. Chua, “Mmgcn: Multi-modal graph convolution network for personalized recommendation of micro-video,” in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 1437–1445.