# Graph Neural Network-Based Knowledge Graph Embedding for Personalized Recommendation Systems

**Isabella Almeida,** ( IsabellaAlmeidaUFF91@hotmail.com )
Universidade Federal Fluminense
**Enzo Miranda** ( enzomiranda93.cos@gmail.com )
Universidade Federal Fluminense
**Bruna Ferreira** ( BrunaFerreira.uff87@hotmail.com )
Universidade Federal Fluminense
**Gael Teixeira** ( gaelteixeirasoc.uff@hotmail.com )
Universidade Federal Fluminense

**Research Article**

**Keywords:** Recommendation systems, Collaborative filtering, Knowledge Graph, Graph Neural Network, Deep learning.

**DOI:** https://doi.org/10.21203/rs.3.rs-2689875/v1

# Graph Neural Network-Based Knowledge Graph Embedding for Personalized Recommendation Systems

Isabella Almeida, Enzo Miranda, Bruna Ferreira, Gael Teixeira,

**Abstract**—With the increase in online content, recommendation systems have become an important tool to filter information overload and provide users with personalized experiences. Collaborative filtering is a classical recommendation technique that represents users and items as vectors. However, recent studies have shown that item attributes, represented as a Knowledge Graph (KG), can provide rich auxiliary information for recommendation systems. This paper proposes a Graph Neural Network (GNN) based on KGs to improve the accuracy, diversity, and interpretability of recommendation systems. The GNN fuses semantic related information in KG to achieve user as well as item representation habits. Neighboring nodes have different weights, which can be aggregated to obtain a more accurate representation of the current entity and improve the accuracy of recommendations. The learning process can be extended to more contents related to user interests, which can improve the diversity of recommendations. This paper proposes a graph neural network based on knowledge graphs, fusing semantic related information in KG to achieve user as well as item representation habits. Neighboring nodes have different weights, which can be aggregated to obtain a more accurate representation of the current entity and improve the accuracy of recommendations. Due to the influence of user preferences, the learning process can be extended to more contents related to user interests, which can improve the diversity of recommendations.

**Index Terms**—Recommendation systems, Collaborative filtering, Knowledge Graph, Graph Neural Network, Deep learning.

## 1 Introduction

With the advancement of Internet technology, people have access to a large amount of online content, such as news, movies, and various products. However, the explosion of online information often overwhelms users, and recommendation systems are an effective information filtering tool that reduces information overload while providing a good user experience. Recommender systems typically search and select a small amount of content to meet the individual interests of users. Collaborative filtering is a classical recommendation technique that represents users and items as vectors. In recent years, deep learning methods have developed rapidly in the fields of natural language, speech, image and video [1]. Using deep learning methods to extract the hidden features of user items from user IDs, review texts, and other data, and to predict the user's rating of new items based on these features to give recommendations is the mainstream practice of traditional recommendation algorithms.

Several recent studies [2, 3, 4, 5] have used item attributes for modeling, and the results of these studies show that attributes are not isolated but are interconnected, one form of which is the Knowledge Graph (KG). KG is a typical heterogeneous network in which each node represents an entity, which can be either an item or an attribute, and two related entities are connected by an edge. KG has rich semantic association information, which can provide rich auxiliary information for recommendation systems. Gori et al [6] first introduced the concept of Graph Neural Network (GCN) by using RNN to compress node information and learn graph node labels. Later, the literature [7] proposed Graph Convolutional Network (GCN), which formally uses CNN to model graph structure data. In this way, GCN can utilize multi-scale information and combine it into higher-level representations. It effectively utilizes graph structure and attribute information, and provides a standard paradigm for migrating other neural networks to graphs in deep learning [8].

KGs have rich semantic association information and can provide rich auxiliary information for recommendation systems. Compared with the recommendation model without KG, the recommendation model using KG can have the following three features [9]: 1) Accuracy. The rich semantic related information can uncover deeper hidden relationships and improve the accuracy of the results. 2) Diversity, the multi-category relationships in KG can expand the user's interests, reasonably spread the recommendation results, and improve the diversity of the recommendation results. 3) Interpretability, KG can connect the user's history of clicks and recommendation records, and provide interpretability for the recommendation system. Although KG has the above advantages, it is still a challenge to use KG for recommendation due to its natural high dimensionality and heterogeneity. One possible approach is KG embedding, which maps entities or relationships in KGs to low-dimensional representation vectors. However, some commonly used KG embedding approaches, such as TransE [10] and TransR [11], focus on modeling semantic

*Isabella Almeida is the corresponding author.

- *Isabella Almeida, Enzo Miranda, Bruna Ferreira, and Gael Teixeira are with University Federal Fluminense, Brazil. (e-mail: IsabellaAlmeidaUFF91@hotmail.com)*

associations and are more suitable for graph applications such as KG complementation and link prediction rather than recommendation [**?**]. In the graph embedding approach, the graph modeling and feature representation are independent from the downstream tasks, and the results obtained from the downstream tasks cannot be used to optimize the feature representation of the graph; moreover, this approach lacks induction capability and requires relearning the feature representation of the entire graph when new nodes are added. To this end, this paper proposes a graph neural network based on knowledge graphs, fusing semantic related information in KG to achieve user as well as item representation habits. Neighboring nodes have different weights, which can be aggregated to obtain a more accurate representation of the current entity and improve the accuracy of recommendations. Due to the influence of user preferences, the learning process can be extended to more contents related to user interests, which can improve the diversity of recommendations.

## 2 Related Work

### 2.1 Deep learning based recommendation models

The first deep learning-based recommendation algorithms were used to extract information from review texts, e.g., ConvMF [12] used CNN to extract user item preferences from reviews, and the performance of the algorithm was greatly improved compared with matrix decomposition methods. Many algorithms around CNNs have emerged since then, for example, DepCoNN [13] uses a parallel CNN structure to simultaneously extract user item preferences. LCPMF combines the LDA topic model in matrix decomposition with CNN to synthesize the topic and deep semantic information of review documents . DIN et al. [14] introduced an attention mechanism to learn user item expressions with good results. MAGN uses a multi-headed attention mechanism, which adds new attention to the attention mechanism to capture the impact that different friends of the user exert on the user in different ways. Wide&Deep [15] consider both low order features and high order features to enhance the effect and use multi-threading to improve the processing efficiency. CapIPTV introduced capsule networks, which exploit the dynamic routing mechanism and clustering features of capsule networks, while using attention mechanism to derive different interest preferences of different users. after CNN, new deep learning methods such as RNN, LSTM [16]. Transformer [17, 18, 19]have emerged using JNTM, RRN, BST and other new recommendation algorithms. After the emergence of graph neural networks, because they can learn the association between users and items that cannot be learned by deep learning methods, graph neural networks and deep learning methods have jointly become the mainstream of recommendation algorithm research.

### 2.2 Graph neural network based recommendation models

Graph neural networks aim to apply neural networks to data in non-Euclidean spaces to perform feature learning. Graph Convolutional Neural Network (GCN) was proposed by Kpif in 2017 [20], which provides a novel idea for processing graph structured data by applying Convolutional Neural Network (CCN) to graph data. The CCN has been applied to graph data. Recently, many research works have applied GCN to recommender systems. PinSAGE [21] applied to the recommendation of bipartite graphs of images and galleries in Pinterest. Monti et al. [22] and Berg et al. [23] modeled the recommender system as a matrix decomposition and designed different GCN models to learn the representation of users and items in the bipartite graph, respectively. Wu et al. [24] used GCNs to learn user/item representations on a user/item structured graph. The difference between these works and the present work is that their models are designed for homogeneous bipartite graphs, which are simpler to model using GCNs because there are no complex relationships. In 2009, the graph neural network (GNN) [25] model was proposed, and in 2017, the emergence of GCN (graph convolutional nerve network) model led to the rapid development of graph nerve networks, which are widely used in the field of recommendation systems. NGCF [26] model introduces GCN into the recommendation algorithm, models the higher-order connectivity of user items and gives recommendations accordingly. KGAT [27] algorithm incorporates attention mechanism into graph neural network and achieves good results. NIA-GCN [28] further considers the interactions between neighbor nodes based on GCN, which can effectively aggregate the information of neighbor nodes at each depth. The GCN-ONCF model designs the GCN as an encoder and transforms the encoding vector into a two-dimensional feature matrix by using the outer product operation, and the convolutional matrix decomposition is realized by the convolutional self-decoder. LightGCN [29] simplifies the redundant part of GCN using bipartite graph based on NGCF, thus improving the model efficiency and performance.

## 3 Methodology

### 3.1 Embedding layer

In a typical recommendation scenario, there will be a set of M users $U = \{u_1, u_2, \ldots, u_M\}$ and a set $V = \{v_1, v_2, \ldots, u_M\}$ containing N items. The user-item interaction matrix is $Y \in R^{M \times N}$ , which reflects the implicit feedback from the user, when $y_{uv} = 1$ means that user u has a connection with item v, such as clicking, browsing or purchasing, and vice versa $y_{uv} = 0$. In addition, there is a knowledge graph G, which consists of entity-relationship-entity triples $(h, r, t)$. Here $h \in E$, $r \in R$ and $t \in E$ denote the head, relationship and tail of the triad, and E and R are the sets of entities and relationships in the knowledge graph, respectively. Given a user-item interaction matrix Y and a knowledge graph $\mathcal{G}$, the goal is to predict whether a user u is potentially interested in an item v or not. That is, the learning function $\hat{y} = F(u, v|Y, \mathcal{G}, \Theta)$, where $\hat{y}$ denotes the probability that user u likes item v and $\Theta$ is a parameter of the function F.

### 3.2 Neighbor sampling

Consider a user u and an item v, v exists as an entity in the knowledge graph, and $N(v)$ is used to denote the set of

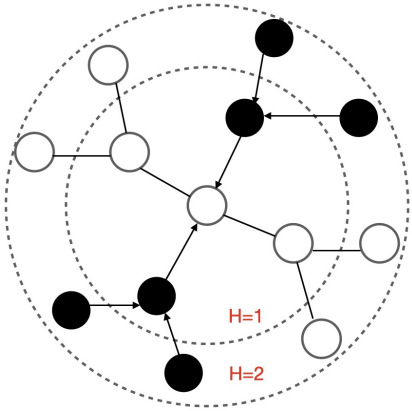


Fig. 1: Aggregation of neighbor nodes after sampling.

neighboring entities of entity v in the knowledge graph r, $e_i$, $e_j$ denotes the relationship between two entities $e_i$ and $e_j$. If user u has ever rated item v, the entity related to the item v is found in the knowledge graph, and then the neighbor node feature aggregation is done for this entity. In the process of neighbor node feature aggregation, an influence factor is added to affect the aggregation result, which can be treated as a measure of user preferences, similar to the attention mechanism.

$$\alpha_r^u = g(u, r), \tag{1}$$

where $u \in R^d$ and $r \in R^d$ represent the characteristics of user u and relationship r, respectively, and d is the dimension of the characteristics. In general, $\alpha_{ur}$ already characterizes the importance of relationship r to user u and can be used to represent the interest preferences of the user.

$$\alpha_e^v = g(v, e), \tag{2}$$

where e is the characteristic of all neighboring nodes of the current entity v. $\alpha_e^v$ portrays the influence of all neighboring entities e on the current entity v.

Each node needs to sample its neighbors to obtain the set of neighbors to be fused before fusing the features of the neighboring entities. Since the neighbor data of an entity in the knowledge graph are not the same, and in an extreme case, the number of neighbors can vary greatly. In order to improve the training efficiency of the model, instead of using all the neighbors of each entity, a set $S(v)$ is sampled uniformly and randomly among all the neighbors, and the number of sampled neighbors is controlled by K. After sampling K neighbor nodes uniformly and randomly, the neighbor nodes of item v are modeled as follows.

$$V_{N(v)}^u = \sum_{e \in N(v)} (\alpha_r^u e + \alpha_e^v e), \tag{3}$$

where e is the feature of the neighbor node of entity v.

### 3.3 Forward propagation layer

Assuming that there are m convolution kernels, each convolution kernel $K \in R^{ks \times ks}$, where ks is the adjustable convolution kernel width $k_i$ degrees, feature extraction for the general preferences of user $u_i$ first goes through the convolution operation.

$$z_{ui} = LeakyReLU(re(b_{ui}) * K_{ki} + \mu_{ui}), \tag{4}$$

where $re()$ operation converts the dimension of $b_{ui}$ into a second-order tensor such that $b_{ui} \in R$. $*$ denotes the convolution operation. $u_i$ is the deviation. After completing one convolution operation the resulting vector $z \in R$, the convolution operation produces m vectors z due to the existence of m convolution kernels. After the convolution operation, each vector $z_{ui}$ of the convolution layer output is subjected to a max pooling operation to further extract the features.

$$o_{ui} = max(z_{ui1}, z_{ui2}, \ldots, z_{uin}), \tag{5}$$

$$O = [o_1, o_2, \ldots, 0_m], \tag{6}$$

After the pooling layer, the resulting features are fed into the fully connected layer. Then, the propagation of the fully connected layer is calculated as follows.

$$p_{ui} = B(m, \beta), \tag{7}$$

$$b_{ui} = LeakyReLU(w_3 \otimes (p_{ui} \odot o) + g_{ui}), \tag{8}$$

### 3.4 Aggregate features

Fuse the current entity feature v with its neighboring features $v_{N(v)}^u$ and use the obtained result as the new feature representation of the current entity.

The GCN aggregator adds two features directly and then uses a nonlinear function to obtain the result.

$$f_{GCN} = LeakyReLU(W \cdot (v + v_{N(v)}^u) + b), \tag{9}$$

where the activation function is set to LeakyReLU. $W \in R^{d \times d}$ is the trainable weight matrix, which is used to extract useful information for propagation operations.

GraphSage aggregator combines two features together and apply a nonlinear transformation to obtain the result.

$$f_{GS} = LeakyReLU(W \cdot concat(v, v_{N(v)}^u) + b), \tag{10}$$

where the activation function is set to LeakyReLU. $W \in R^{d \times d}$ is the trainable weight matrix.

### 3.5 Scoring prediction layer

After forward propagation layer, we can obtain the association relationship feature vector $e_{ui}, e_{ij} \in R^d$ and general preference feature vector $b_{ui}, b_{ij} \in R^d$ for user $u_i$ and product $i_j$. In this paper, we use concat method to fuse the features of the two feature vectors to obtain the final feature representation $U_i, I_j \in R^{2d}$ for user $u_i$ and product $i_j$.

$$U_i = e_{ui} \oplus b_{ui}, \tag{11}$$

$$I_j = e_{ij} \oplus b_{ij}, \tag{12}$$

The final score prediction of the product $i_j$ by the user $u_i$ is as follows.

$$P_{ij} = U_i \otimes I_j^T, \tag{13}$$

### 3.6 Optimization

In order to learn the model parameters so that the model can better model the user and item characteristics, this paper uses the BPR loss, which is widely used in recommender systems. The BPR loss is based on Bayesian ranking, which takes into account the relative order of observable and unobservable user-item interactions, and considers the observed interaction items to be more important than the unobservable ones.

$$Loss = \sum_{(u,a,b)\in D} -ln\theta(P_{ua} - P_{ub}) + \lambda||\Theta||_2^2, \quad (14)$$

where $R^+$ represents the positive sample, i.e. the observed interaction term. $R^-$ represents the negative sample, i.e., the unobserved interaction term. $\theta$ represents the $sigmod()$ function. $\lambda||\Theta||_2^2$ is the regular term. $\lambda$ is the adjustable coefficient of the regular term, which is used to control the size of the model parameters to prevent overfitting. In this paper, the model network is trained by stochastic gradient descent, and the Adam optimizer is used to optimize the model parameters and minimize the loss function.

## 4 EXPERIMENTS

### 4.1 Dataset

Two datasets were selected for the experimental part: MovieLens-1M and Amazon. MovieLens-1M is a widely used benchmark dataset for movie recommendations, which contains explicit user ratings of movies from 1 to 5 on the MovieLens website. The Amazon dataset, one of the most widely used datasets for recommendation systems, contains purchase records for 29 categories of products from 196 to 2018, including product IDs, user IDs, review texts, product tags, and many other data information.

### 4.2 Baselines

- LFM is the most classical matrix decomposition algorithm, which only uses the interaction between users and products to learn the implicit features of users and products, and uses product ratings as the prediction target for learning, and its recommendation effect is affected by data sparsity.
- DeepFM [30] is a modification of Wide&Deep, which is obtained by training a shallow model and a deep model. It consists of two parts, the neural network module and the factorizer module, which are responsible for the extraction of low-order features and high-order features, respectively, and the input of both parts is the same as LFM, both are raw features.
- NGCF follows the standard GCN model and has successfully applied GCN in recommendation systems. It successfully uses dichotomous graphs to extract the embedded interactions (i.e., associations) between users and products to obtain rating predictions.
- LightGCN removes the redundant feature transformation and activation function parts based on NGCF, which greatly improves the operational efficiency and prediction accuracy of the model.
- MMGCN [31] is a multimodal fused graph convolutional network. MMGCN can not only make use of multimodal dependencies effectively, but also leverage speaker information to model inter-speaker and intra-speaker dependency.

### 4.3 Hyperparameter adjustment

The regularization weights are adjusted using the control variables method, where the number of graph convolution iterations la=3, the dimension of the embedding vector d=64. The analysis in Table 4 shows that the model is most effective when the parameters are 2E-5 to 1E-5. It is noted that the effect of the model recall decreases from 5E-5 to 2E-5 in the data set SO, and the regularization weight is 5E-5 considering the generalization performance of the model. In the comparison experiments for the adjustment of the number of iterations of the graph convolution, the embedding vector dimension d = 64. It is obvious from the analysis of the data of the comparison experiments that the model results are optimal at la = 2, so the final number of iterations is determined as 2. Considering the sparsity of the dataset, it can be judged that the vast majority of nodes in the graph neural network constructed in each of the four datasets have 2-hop to 3-hop paths, and a small number of nodes have 3-hop or more paths (which can be verified by the dataset). In this case, more correlations can be extracted at la = 2 than at la = 1, and the model effect will be relatively better. la = 3 does not extract more correlations than la = 2, so the model effect will not be better, but will be relatively lower. The analysis shows that the model effect increases as the dimensionality of the embedding vector increases. It is noted that the average growth rate of the model effect in the four data sets of d=16 32 and 64 is larger than that of d=64 128. Considering that the space occupied by the 64-dimensional embedding vector doubles to 128-dimensional embedding vector, the growth rate of the model effect does not reach the expected growth rate, so the embedding vector dimension is finally adopted as 64-dimensional.

### 4.4 Metrics

This paper adopts the topK recommendation method for recommendation, where K=20. Two evaluation metrics, recall and normalized discounted cumulative gain (NDCG), are used to evaluate the model performance. Recall is used to measure the proportion of the number of products in the top20 recommendation list that users have interacted with to the number of products in the test set that users have interacted with, and the higher the recall rate, the better the model performance. Assuming that the set of all users in the test set is U, for any user $u \in U$, the recommendation list of top20 is $L_u$, and the real interaction list of user u in the test set is L, then the model recall is calculated as follows.

$$recall@20 = \sum_{u\in U} \frac{1}{|U|} \frac{|L_u \cap L_u^{test}|}{|L_u|^{test}}, \quad (15)$$

NDCG measures the relevance of the recommendation results in different positions in the recommendation list. The NDCG evaluates the recommendation quality of the recommendation list of all users. Assuming Li is the recommendation at position i in the top20 recommendation list, $f(x)$ takes 1 when $x > 0$, otherwise it takes 0. Assuming

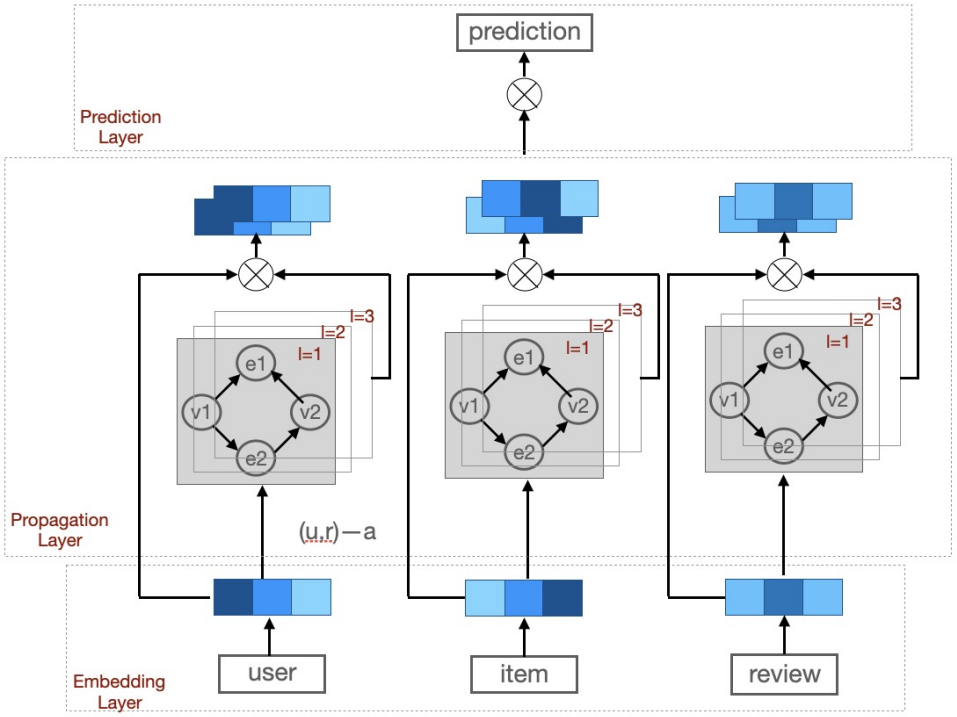


Fig. 2: Modelstructure.

the relevance score $rel \in \{0, 1\}$, the NGCD calculation formula is as follows.

$$NDCG@20 = \sum_{u \in U} \frac{1}{|U|} \frac{DCG_u@20}{IDCG_u@20} \tag{16}$$

### 4.5 Result analysis

In this paper, five models were experimentally compared on two datasets, and the comparison results are shown in Tables 2 and 3. The LMF model performs poorly in both datasets, as both datasets are sparse by one thousandth of a percent, suggesting that matrix decomposition is not sufficient to better capture user and product characteristics in extremely sparse datasets. From the comparison between our model and other baselines, we can see that our model outperforms the other models in both datasets, with at least 0.16% improvement in recall and 0.41% improvement in NDCG. Compared with the suboptimal LightGCN model, the effect of LightGCN is improved by about 4.3%, which proves that the model design method of this paper contributes to the improvement of the model effect. The LightGCN model removes the redundant feature transformations and nonlinear transformations in the graph convolutional neural network part compared with NGCF. From the experimental results, the effect of the LightGCN model is improved by about 21% compared with that of NGCF, and the improvement is obvious, which shows the effectiveness of the LightGCN improvement. The NGCF model introduces GCN into the recommendation algorithm, and from the experimental results, the model effect is improved by about 17% compared with the NeuMF model, which shows that the graph neural network has the effect of alleviating data sparsity for extremely sparse data sets and contributes to the improvement of the recommendation effect.

## 5 Conclusion

Recommender systems typically search and select a small amount of content to meet the individual interests of users. Collaborative filtering is a classical recommendation technique that represents users and items as vectors. In recent years, deep learning methods have developed rapidly in the fields of natural language, speech, image and video. Using deep learning methods to extract the hidden features of user items from user IDs, review texts, and other data, and to predict the user's rating of new items based on these features to give recommendations is the mainstream practice of traditional recommendation algorithms. KG is a typical heterogeneous network in which each node represents an entity, which can be either an item or an attribute, and two related entities are connected by an edge. KGs have rich semantic association information and can provide rich auxiliary information for recommendation systems. Compared with the recommendation model without KG, the recommendation model using KG can have the following three features. Accuracy. The rich semantic related information can uncover deeper hidden relationships and improve the accuracy of the results. Diversity, the multi-category relationships in KG can expand the user's interests, reasonably spread the recommendation results, and improve the diversity of the recommendation results. Interpretability, KG can connect the user's history of clicks and recommendation records, and provide interpretability for the recommendation system. Although KG has the above advantages, it is still a challenge to use KG for recommendation due to its natural high dimensionality and heterogeneity. One possible approach is KG embedding, which maps entities or relationships in KGs to low-dimensional representation vectors. However, some commonly used KG embedding approaches, such as TransE and TransR, focus on modeling semantic associations

TABLE 1: Statistical information of datasets.

| dataset | Movielens | Amazon |
|---|---|---|
| **users** | 6036 | 26988 |
| **items** | 2347 | 14240 |
| **interactions** | 753772 | 417492 |
| **entities** | 7008 | 25787 |

TABLE 2: Model precision comparison on Movielens-1m.

| Model | recall | NDCG |
|---|---|---|
| **LFM** | 5.38 | 4.55 |
| **DeepFM** | 5.81 | 4.82 |
| **NGCF** | 6.76 | 6.17 |
| **LightGCN** | 7.22 | 6.68 |
| **MMGCN** | 7.57 | 6.93 |
| **Ours** | 7.85 | 7.01 |

TABLE 3: Model precision comparison on Amazon.

| Model | recall | NDCG |
|---|---|---|
| **LFM** | 4.28 | 4.03 |
| **DeepFM** | 4.76 | 4.22 |
| **NGCF** | 5.37 | 4.96 |
| **LightGCN** | 5.89 | 5.61 |
| **MMGCN** | 6.03 | 6.33 |
| **Ours** | 6.14 | 6.45 |

and are more suitable for graph applications such as KG complementation and link prediction rather than recommendation. In the graph embedding approach, the graph modeling and feature representation are independent from the downstream tasks, and the results obtained from the downstream tasks cannot be used to optimize the feature representation of the graph; moreover, this approach lacks induction capability and requires relearning the feature representation of the entire graph when new nodes are added. To this end, this paper proposes a graph neural network based on knowledge graphs, fusing semantic related information in KG to achieve user as well as item representation habits. Neighboring nodes have different weights, which can be aggregated to obtain a more accurate representation of the current entity and improve the accuracy of recommendations. Due to the influence of user preferences, the learning process can be extended to more contents related to user interests, which can improve the diversity of recommendations.

## REFERENCES


[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 2. IEEE, 2005, pp. 729–734.

[7] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[8] Y. Wei, X. Wang, W. Guan, L. Nie, Z. Lin, and B. Chen, "Neural multimodal cooperative learning toward micro-video understanding," *IEEE Transactions on Image Processing*, vol. 29, pp. 1–14, 2019.

[9] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[10] 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.

[11] 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.

[12] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *Proceedings of the 10th ACM conference on recommender systems*, 2016, pp. 233–240.

[13] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," in *Proceedings of the tenth ACM international conference on web search and data mining*, 2017, pp. 425–434.

[14] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 1059–1068.

[15] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st workshop on deep learning for recommender systems*, 2016, pp. 7–10.

[16] 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.

[17] C. Yang, M. Sun, W. X. Zhao, Z. Liu, and E. Y. Chang, "A neural network approach to jointly modeling social networks and mobile trajectories," *ACM Transactions on Information Systems (TOIS)*, vol. 35, no. 4, pp. 1–28, 2017.

[18] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, "Recurrent recommender networks," in *Proceedings of the tenth ACM international conference on web search and data mining*, 2017, pp. 495–503.

[19] Q. Chen, H. Zhao, W. Li, P. Huang, and W. Ou, "Behavior sequence transformer for e-commerce recommendation in alibaba," in *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*, 2019, pp. 1–4.

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

[21] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 974–983.

[22] F. Monti, M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," *Advances in neural information processing systems*, vol. 30, 2017.

[23] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.
[24] Y. Wu, H. Liu, and Y. Yang, "Graph convolutional matrix completion for bipartite edge prediction." in *KDIR*, 2018, pp. 49–58.
[25] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.
[26] 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.
[27] 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.
[28] J. Sun, Y. Zhang, W. Guo, H. Guo, R. Tang, X. He, C. Ma, and M. Coates, "Neighbor interaction aware graph convolution networks for recommendation," in *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, 2020, pp. 1289–1298.
[29] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 639–648.
[30] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: a factorization-machine based neural network for ctr prediction," *arXiv preprint arXiv:1703.04247*, 2017.
[31] 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.