Heterogeneous Graph Recommendation Model based on Graph Neural Network

Van Norton
University of Canberra
Mark Miller
University of Canberra
Pierre Meyer (PierreMeyer.unistra@hotmail.com)
University of Strasbourg
John Ward
University of Canberra
Gregory Agnew
University of Canberra

Research Article

Keywords: Recommendation, Recommender System, GNN, GCN, Graph

Posted Date: September 27th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2102785/v1

Heterogeneous Graph Recommendation Model based on Graph Neural Network

Van Norton, Mark Miller, Pierre Meyer, John Ward, Gregory Agnew,

Abstract—With the development of the Internet in recent years, the network information is exploding, people have entered the era of big data, from the lack of information in the past to the great information overload nowadays. It has become an important challenge to filter the information that is beneficial to you in the ocean of information on the Internet. As a subset of the information filtering system, the recommendation system mainly relies on the historical interaction records of users' products and its own attribute information to explore the potential preferences and needs of users, which greatly reduces the time for users to filter information and is helpful for improving user experience and alleviating the information overload problem. Traditional recommendation algorithms represented by collaborative filtering often face the cold-start problem. They usually recommend products for users based on their purchase history or rating information of products, however, the performance of such recommendation algorithms decreases dramatically when the interaction history of users' products is sparse. Heterogeneous graph networks contain multiple types of nodes and edges due to their inclusion. It contains richer semantic information, and more general recommendation algorithms based on heterogeneous graphs are relatively less studied. Therefore, the purpose of this paper is to study graph neural network-based recommendation algorithms, mainly considering the heterogeneity of the network structure and the interaction order information between nodes, and then build a new recommendation model to cope with the cold start problem and improve the recommendation effect.

Index Terms—Recommendation, GNN, Information.

1 Introduction

With the development of the Internet in recent years, the network information is exploding, people have entered the era of big data, from the lack of information in the past to the great information overload nowadays. It has become an important challenge to filter the information that is beneficial to you in the ocean of information on the Internet. As a subset of the information filtering system, the recommendation system mainly relies on the historical interaction records of users' products and its own attribute information to explore the potential preferences and needs of users, which greatly reduces the time for users to filter information and is helpful for improving user experience and alleviating the information overload problem [1] . Research on personalized recommendation algorithms started in the early 1990s and has been widely used in various fields such as news recommendations and e-commerce recommendations [2]. Early work mainly took the collaborative filtering approach [3]. Collaborative filtering is based on the assumption that users with similar historical interaction behaviors may have similar preferences for products. Yehuda Koren et al. [4] proposed a matrix decomposition-based recommendation algorithm that decomposes the rating matrix into implicit feature representations of users and goods to explore deeper relationships between users and goods. Unlike matrix decomposition that multiplies the implicit feature representations of users and goods directly to model the user-good interaction matrix, Xue et al. [5] proposed a deep matrix decomposition model that uses a nonlinear multilayer perceptron to replace direct multiplication, which can better capture the nonlinear relationships. However, most existing collaborative filtering algorithms only model user and commodity representations through user- and commodity-specific attributes without considering the potential higher-order relationships between user-commodity interaction graphs. He et al. [6] proposed a collaborative filtering algorithm based on neural graphs, which propagates node representations on the constructed user-commodity bipartite graph by mining higher-order structural information.

Traditional recommendation algorithms represented by collaborative filtering often face the cold-start problem. They usually recommend products for users based on their purchase history or rating information of products, however, the performance of such recommendation algorithms decreases dramatically when the interaction history of users' products is sparse. The cold-start problem can be simply described as how to provide recommendations for new users or new items that join the system. Imagine a scenario where a new user joining a social platform has a very limited interaction history within the platform, which makes it difficult for a similarity-based collaborative filtering recommendation algorithm to cope. The lack of interaction information makes it difficult for collaborative filtering to identify other users with similar interests to the new user, and therefore to recommend appropriate items for that user. This is one form of the cold-start problem: user cold-start [7, 8]. The counterpart is item cold start: when a new item is available on the shelf, collaborative filtering has difficulty in pushing it to the right target users in the short term [9]. In addition, there is the more extreme system cold start, i.e., how to provide

*Van Norton is the corresponding author.

- *Van Norton, Mark Miller, John Ward, Gregory Agnew, are with University of Canberra, Australia. Pierre Meyer are with University of Strasbourg, France. (e-mail: PierreMeyer.unistra@hotmail.com).*

a timely and available recommendation service on a newly developed platform.

For non-Euclidean data like heterogeneous graphs, properties such as different graph sizes and node disorder make traditional deep learning models (RNNs, CNNs) cannot be applied directly to extract effective representations, while graph neural networks can be applied to non-Euclidean domains like graphs by aggregating information from nearest neighbor nodes and performing convolution directly on the graph structure to effectively capture dependencies and higher-order representations between data, and thus have received wide attention from researchers in recent years [10, 11, 12].

In terms of recommendation based on graph neural networks, Ying et al. [13] with 2018 proposed PinSage, the first application of GNN in the field of commercial recommendation, which achieved good results. Wang et al. [14] proposed KGCN to apply knowledge graphs to the recommendation field while combining graph neural networks to mine the relationship of commodities on the attributes of knowledge graphs, effectively capturing The inner connection of products is effectively captured. Subsequently, Wang et al. [15] proposed KGAT to incorporate an attention-based aggregation approach to solve the problem of higher-order relationship inequality. Heterogeneous graph networks contain multiple types of nodes and edges due to their inclusion. It contains richer semantic information, and more general recommendation algorithms based on heterogeneous graphs are relatively less studied. Therefore, the purpose of this paper is to study graph neural network-based recommendation algorithms, mainly considering the heterogeneity of the network structure and the interaction order information between nodes, and then build a new recommendation model to cope with the cold start problem and improve the recommendation effect.

2 Related Work

2.1 Graph Neural Network Model

2.1.1 Convolutional Graph Neural Network

Convolutional graph neural networks are a class of graph neural networks that define convolutional operations on the graph structure. Given an undirected connected graph $\mathcal{G} = (\sqsubseteq, \epsilon)$, where $\sqsubseteq$ denotes the set consisting of all nodes within this graph, and ϵ denotes the set of all connected edges. And given node $u, v \in \sqsubseteq$. When there is a connected edge between node u and v, we use $(u, v) \in \epsilon$ to denote this connected edge. In addition, the u,v node has an initial content feature x_u, x_v, which is generated by collecting the attribute description fields of the node. We use $N(u) = \{v | v \in \sqsubseteq, (u, v) \in \epsilon\}$ to denote the set consisting of the neighboring nodes of node u, or the neighborhood of u. The key idea of convolutional neural networks is the way by which the initial features from node u, and other information on the graph to obtain the representation of node u. The idea can be broadly divided into three steps.

- Node sampling. In order to obtain a representation of node u, it is first necessary to sample the neighborhood of u in $\mathcal{G}$, i.e., to determine N.
- Neighborhood aggregation. The features of the nodes in the sampled neighborhood N are collected and aggregated in some way, which we call aggregation.
- Node update. The update yields the hidden features of the next level of u. The essential difference between different convolutional graph neural is the selection of functions. In general, convolutional graph neural networks extract high-level node representations by overlaying multiple graph convolutional, or message passing, layers, each with different weights and generally without interdependencies. The number of layers in a convolutional graph neural network is generally a predetermined fixed value, and the model superimposes k layers of graph convolutional layers to transform the initial features of node v to the output features step by step. The popularity of convolutional graph neural networks has grown rapidly in recent years due to the high availability and effectiveness of graph convolution and the greater ease of combination with other neural networks.

Based on the differences in the definition and selection of the functions mentioned above, convolutional graph neural networks can be divided into two main categories: spectral-domain graph convolutional neural networks and spatial-domain graph convolutional neural networks [16, 17]. The difference between them is that the spectral domain approach defines graph convolution as the filtering of signals on the graph starting from the spectral graph theory direction [18], while the spatial domain approach defines graph convolution as the aggregation of signals from nodes and their neighboring nodes starting directly from the node domain direction. The idea of the spectral domain approach originated in the field of signal processing [18, 19, 20]. Since it is difficult to guarantee the translation invariance of traditional convolution in the graph structure, the spectral domain approach shifts the message passing from the node domain to the spectral domain to circumvent the translation invariance limitation. In 2016, Kipf et al. [21] proposed the graph convolutional neural net (GCN) to build a bridge between the spectral domain-based approach and the spatial domain-based approach. The spatial domain-based graph convolution method defines convolution directly on the node domain. The spatial domain convolution is more intuitive compared to the spectral domain, i.e., the node messages are passed directly on the connected edges of the graph. Graph structured neural networks construct messages by combining neighborhood features at all levels, i.e., they are not limited to messages at the previous level [22]. Also some work in the neighborhood sampling step is not limited to such a simple definition equation . GraphSage [11], which performs random sampling of the nodes close to node u, by limiting the neighborhood size in order to achieve inference capabilities in the big graph. This approach also provides the ability to represent nodes outside the graph, enabling inductive inference.

2.1.2 Graph Auto Encoder

Autoencoders (AEs) are a type of neural network that consists of an encoder and a decoder which are composed of an encoder and a decoder. The encoder obtains a low-dimensional vector representation of the original data

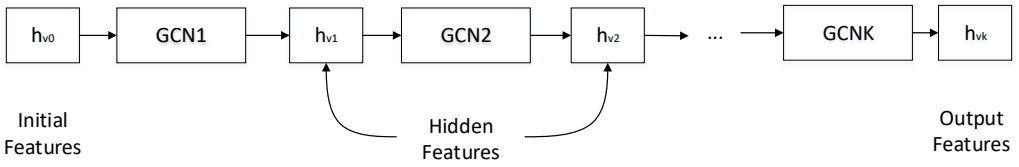

Fig. 1: Convolutional graph neural network structure.

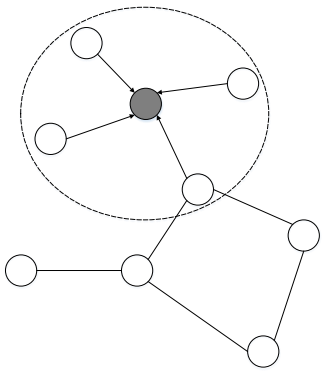

Fig. 2: Schematic of spatial domain-based graph convolution.

through the neural network, and the decoder reduces the low-dimensional vector representation to the original data through the neural network. If we consider the decoder as a generative model, we can use this generative model to obtain approximately real samples as long as we have a low-dimensional vector representation. The graph autoencoder is designed to encode the graph to obtain the embedded vector by using this feature of autoencoder and reconstruct the graph structure by decoding to optimize the learned hidden vectors [23]. In general, the focus of graph autoencoders is not on reconstruction, but on embedding, i.e., how to efficiently map graph topological information to a low-dimensional vector space i.e., how to efficiently map graph topological information to a low-dimensional vector space. The performance of the embedding results will directly affect the performance of the downstream tasks of the network [24, 25]. The Variational Graph Auto-Encoder (VGAE) proposed by Kipf et al. [26] combined graph auto-encoders with Variational Auto Encoders (VAEs), which instruct the encoder to encode the sample mean and variance information to learn the sample distribution of the samples.

2.2 Recommendation Algorithms

A complete recommendation system consists of user resources, item resources and recommendation algorithms. The recommendation algorithm is the most important part of the recommendation system [27], which needs to analyze the user and item resources, establish the corresponding user-item interaction model, and explore the deep connection to achieve the recommendation task. Traditional recommendation methods can be divided into three main categories: content-based recommendation, collaborative filtering-based recommendation, and Hybrid recommendation.

Collaborative Filtering (CF) is one of the most widely used and successful recommendation system techniques. Traditional collaborative filtering recommendation algorithms are based on collaborative filtering of nearest neighbors and model-based collaborative filtering. The Collaborative filtering based on nearest neighbors finds the nearest neighbor relationship by analyzing the similarity between users and users or items and items. This category can be broadly divided into user-based collaborative filtering and item-based collaborative filtering. collaborative filtering based on users and items [28]. Specifically, the general steps are: 1)Calculate the similarity of target users to other users. 2)Identify neighboring users of the target user. 3) Predictive scoring. There are two main methods of providing recommendations to target users: predictive scoring and providing a list of recommendations for the top N items.

2.3 Recommendation Algorithm based on Graph Neural Network

Currently, more and more researchers are focusing on applying graph neural network technology to the field of recommendation systems, and many recommendation algorithms based on graph neural networks have been born [29].

- Graph Convolutional Matrix Complementation (GC-MC) [30]. GC-MC aims at solving the scoring prediction problem, specifically described mathematically as solving the matrix complementation problem in terms of GAE. GC-MC does not use additional information (i.e., content features of users and items) and only predicts by learning the topological connectivity of the graph. In addition, GC-MC does not consider the node's own representation in the message update, but only uses the representation obtained from the neighborhood aggregation for the update.
- Stacked Reconstructive Graph Convolutional Graph Neural Network (STAR-GCN) [31]. the core of STAR-GCN is the stacking of multiple coding blocks consisting of GCN layers, each with the same structure, e.g., GC-MC can be chosen as the coding block. The motivation for stacking coding blocks instead of directly stacking multiple GCN layers is that stacking too many convolutional layers may introduce the problem of "over-smoothing".
- Graph Neural Network Collaborative Filtering (NGCF) [32]. the core idea of NGCF is that messages on graphs will not only be passed on the graph topology but also on the implicitly connected edges of nodes with similarity. Moreover, since NGCF chooses Residual Network (ResNet) as the backbone network, then it

uses different levels of representation to obtain the final node embedding. The motivation is that: 1) item representations that match the user's interest should be delivered to the user more often. 2) the representations obtained in different layers will reinforce the messages that are delivered on different connections.

3 METHODOLOGY

In this paper, we propose a new heterogeneous graph neural network recommendation model. Specifically, we first construct a heterogeneous user-item-topic graph to explicitly model the interaction between users, item and topics. Topic information can help better reflect users' interests and alleviate the sparsity problem of user-item interactions. To encode higher-order relationships between users, items and topics, this paper uses graph neural networks (GNN) to propagate feature representations over graphs to learn user and item representations, and user embeddings learned from complete user click histories of heterogeneous graphs can capture users' interests.

3.1 Symbols and Definitions

There are click histories of K users $U = \{u_1, u_2, \ldots, u_k\}$ and M items $I = \{i_1, i_2, \ldots, i_m\}$. Based on the implicit feedback from the user there is a user-item interactions matrix $Y \in \mathbb{R}^{|u|\times|I|}$, where y=1 means that user w clicked on item i otherwise y=0. From the timestamped click history, we can obtain the sequence $S_u = \{i_{u,1}, i_{u,2}, \ldots, i_{u,n}\}$ of recent clicks for a particular user u, where $i_{u,j}$ is the j-th item clicked by user u. Given Y and S_u, pur goal is to predict whether a user u is potentially interested in an item i that he/she has not seen before.

3.2 Information Extraction

We use two parallel CNNs as information extractors to learn the title-level and summary-level representations of items, using the title and summary of item as input, respectively. These two representations are concatenated as the final text feature representation. Specifically, the title can be denoted as $T = [w_1, w_2, \ldots, w_m]^T$, and the summary can be denoted as $P = [e_1, f(c_1), e_2, f(c_2), \ldots, e_n, f(c_n)]^T$.

The title T and the summary P are fed into two parallel CNNs with independent weight parameters, respectively. Thus we obtain their feature representations $\hat{T}$ and $\hat{P}$ respectively by these two parallel CNNs. Finally, we concatenate $\hat{T}$ and $\hat{P}$ together as the final news text feature representation as the following equation:

$$d = f_c([\hat{T}, \hat{P}]), \tag{1}$$

where $d \in \mathbb{R}^D$, f_c is a fully connected layer.

3.3 User Interest Modeling

We first construct a user-item-summary heterogeneous graph containing the user's complete history of clicks. The introduced summary information helps to better indicate the user's interest and mitigate the sparsity of user-item interactions. We then use graph convolutional networks to propagate node features over the graph to learn embedding representations of users and items to encode higher-order information between users and items. In this paper, we construct a heterogeneous undirected graph $\mathcal{G}$, where V and R are the set of nodes and the set of edges, respectively. This graph contains three types of nodes: users, items and summaries.

If user u clicks item i, a user-item edge is established, i.e., $y = 1$. For each item i, this algorithm can obtain its summary distribution by LDA, and then connect i with the maximum probability summary z to establish the connection.

$$\theta_u = \{\theta_{u,i}\}_{i=1,i=2,\ldots,i=k} \sum_{i=1}^{k} \theta_{u,i} = 1, \tag{2}$$

GNN is used to capture the high order relationship between users and items through propagation embedding.

$$h_{Nv} = aggregate(\{W^t h_u^t, u \in N_v\}), \tag{3}$$

$$h_v = \delta(W \cdot h_{Nv} + b), \tag{4}$$

where aggregate is the aggregation function, which aggregates information from neighboring nodes. In this paper, we use the mean aggregation function, which simply takes the average of the vectors of neighboring nodes. Nv denotes the set of neighbors of a particular node v, and W^t is a learnable transformation matrix that transforms different types of nodes h_u^t into the same space. W and b are the weight matrix and bias of a GNN layer for updating the embedding vector of the central node h_v. Consider user u and item i candidate pairs, and we use inter-U and Z to denote respectively the set of users and summaries directly connected to i directly. For the topology of item i, we first compute the linear average combination of all its sampled neighbors.

$$i_N = \frac{1}{|S(i)|} \sum_{u \in S(i)} W_u u + \frac{1}{|Z(i)|} \sum_{z \in Z(i)} W_z z, \tag{5}$$

where $u \in \mathbb{R}^D$ and $z \in \mathbb{R}^D$ are the embedding representations of the adjacent users and summary of item i, respectively. u and z are initialized randomly, while i is initialized using the feature embeddings obtained by the information extractor. Using the neighborhood representation i update the candidate item representation.

$$\hat{i} = \theta(W^1 \cdot d_N + b^1), \tag{6}$$

where θ is the nonliner ReLU. W^1,b^1 are the variational matrix and bias of the first layer of GNN, respectively.

With GNN, we can obtain end-user and news embeddings u,$\hat{d}$ that have been encoded with higher-order information. user embeddings learned through the complete user click history should be able to capture relatively stable user interests.

Given a user u and his/her most recently clicked l news items, we use an attention mechanism to model the differential impact of a user's most recently clicked item on a candidate item i.

$$u_j = tanh(W d_j + b), \tag{7}$$

$$u = tanh(W d + b), \tag{8}$$

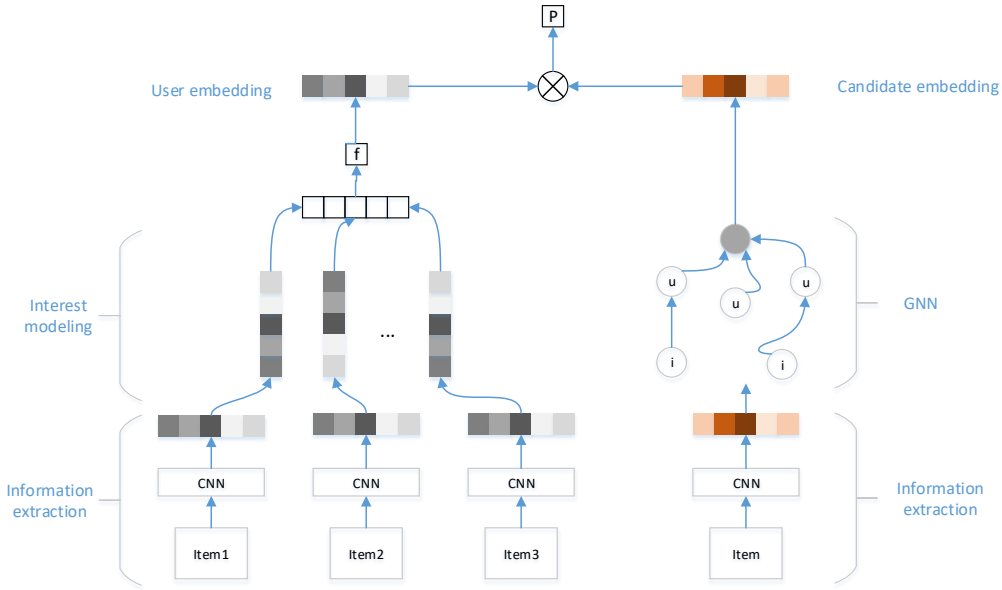

Fig. 3: Schematic diagram of our model.

$$\alpha_j = \frac{exp(v^T(u+u_j))}{\sum_j exp(v^T(u+u_j))}, \tag{9}$$

$$u_c = \sum_j \alpha_j i_j, \tag{10}$$

where u_c is the current content-based interest representation of the user, α is the influence size, and D is the dimensionality of the embedding vector.

3.4 Prediction and Model Optimization

The user embedding u and the candidate item embedding i are fed into a fully connected layer to predict the probability of user u clicking on the item i:

$$\hat{y} = DNN(u, \hat{i}), \tag{11}$$

We use cross-entropy as the loss function.

$$\mathcal{L} = -\{\sum ylog\hat{y} + (1-y)log(1-\hat{y})\} + \lambda||W||, \tag{12}$$

where $||W||$ is the L2 regularization of all trainable parameters and λ is the penalty weight. In addition, we use dropout and early stop to avoid overfitting.

4 EXPERIMENTS

4.1 Datasets

To verify the effectiveness of our method, we conducted extensive experiments on three real datasets. The statistical properties of the datasets are summarized in Table 1.

Movielens is a benchmark dataset that has been widely used for recommendation tasks. In our experiments, we used a version with 100K interaction records and 1M interaction records. The category information of movies is introduced as a item attribute.

Yelp is a business recommendation dataset that records user ratings of local businesses. In this dataset, we use the category of the business as the item attribute information to enrich the user-item interaction graph.

4.2 Metrics

In our experiments, for each user, the last interacting item is selected as the test and the remaining data is used for training. In the test set, we randomly select 100 items that do not interact with the user as negative samples and rank the 100 sampled items from high to low scores. For a fair comparison with the comparison methods, the same set of negative samples is used for each (user, item) pair in all method test sets. We use two popular metrics, HR and NDCG, to evaluate recommendation performance.

$$HR@K = \frac{1}{||}\sum_{u\in U}\delta(p_u \in R_u^k), \tag{13}$$

$$NDCG@K = \frac{1}{||}\sum_{u\in U}\frac{1}{(log_2(index_u+1))}, \tag{14}$$

where $K \in [5, 10, 15, 20]$. p_u is the corresponding true score in the test set. The higher the evaluation index score, the better the performance of the recommendation.

4.3 Baselines

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

- GC-MC [30]. GC-MC is a classic work in the field of graph neural networks for recommendation algorithms, which achieves an embedding representation of nodes on the graph by imposing a graph self-encoder on the user-item bipartite graph.
- NGCF [32]. A graph-based collaborative filtering model that exploits the higher-order connectivity in user-goods interaction graphs to efficiently inject collaborative signals into the representation learning process in an explicit manner.
- STAR-GCN [31]. GC-MC can be chosen as the coding block. The motivation for stacking coding blocks instead of directly stacking multiple GCN layers is that stacking

TABLE 1: Statistical information of datasets.

dataset	users	items	categories	interactions
Movielens 1M	6040	3260	18	165.3
Movielens 100k	943	1152	19	103.9
Yelp	10002	10373	58	10

TABLE 2: Model Performance Comparison on Movielens-1M.

Model	HR@5	NDCG@5	HR@10	NDCG@10	HR@15	NDCG@15	HR@20	NDCG@20
GC-MC	0.434	0.356	0.513	0.432	0.617	0.508	0.728	0.532
STAR-GCN	0.453	0.372	0.522	0.456	0.713	0.529	0.733	0.558
NGCF	0.476	0.401	0.587	0.457	0.748	0.559	0.746	0.589
OURS	0.511	0.422	0.598	0.503	0.607	0.532	0.767	0.602

TABLE 3: Model Performance Comparison on Movielens-100K.

Model	HR@5	NDCG@5	HR@10	NDCG@10	HR@15	NDCG@15	HR@20	NDCG@20
GC-MC	0.421	0.335	0.483	0.392	0.503	0.411	0.634	0.507
STAR-GCN	0.433	0.367	0.465	0.387	0.508	0.490	0.656	0.553
NGCF	0.451	0.389	0.543	0.423	0.565	0.501	0.705	0.573
OURS	0.498	0.412	0.565	0.483	0.596	0.503	0.731	0.596

TABLE 4: Model Performance Comparison on Yelp.

Model	HR@5	NDCG@5	HR@10	NDCG@10	HR@15	NDCG@15	HR@20	NDCG@20
GC-MC	0.347	0.305	0.356	0.322	0.365	0.334	0.387	0.345
STAR-GCN	0.355	0.323	0.378	0.341	0.392	0.364	0.400	0.351
NGCF	0.372	0.336	0.380	0.345	0.395	0.377	0.413	0.375
OURS	0.408	0.349	0.417	0.362	0.435	0.387	0.453	0.398

too many convolutional layers may introduce the problem of ”over-smoothing”.

4.4 Parameter Setting

We implement the proposed model based on the Tensorflow deep learning framework. The embedding vector dimension is set to 64. for MovieLens and Yelp, the hyperparameters for balancing the user's dynamic interest and static interest weights are set to 0.5 and 0.2, respectively. in addition, the learning rate for MovieLens is 0.0005 and for Yelp is 0.00005. For all models, we apply a grid search to traverse the hyperparameters.

4.5 Result Analysis

Our proposed model consistently outperforms all comparison methods on all datasets, including two dense datasets (i.e., ML100K and ML1M) and one sparse dataset (i.e., Yelp). These results validate that our model can effectively model users and items in both sparse and dense datasets due to the full consideration of higher-order heterogeneous co-information. Recommendation models based on heterogeneous information networks generally outperform traditional collaborative filtering methods. In particular, there is a great improvement on sparse datasets (i.e., Yelp), indicating that using heterogeneous information networks to fuse auxiliary information for recommendation can alleviate the problem of dataset sparsity and improve recommendation performance.

5 Conclusion and Future Work

Research on personalized recommendation algorithms started in the early 1990s and has been widely used in various fields such as news recommendations and e-commerce recommendations. Early work mainly took the collaborative filtering approach. Collaborative filtering is based on the assumption that users with similar historical interaction behaviors may have similar preferences for products. Traditional recommendation algorithms represented by collaborative filtering often face the cold-start problem. They usually recommend products for users based on their purchase history or rating information of products, however, the performance of such recommendation algorithms decreases dramatically when the interaction history of users' products is sparse. The cold-start problem can be simply described as how to provide recommendations for new users or new items that join the system. Imagine a scenario where a new user joining a social platform has a very limited interaction history within the platform, which makes it difficult for a similarity-based collaborative filtering recommendation algorithm to cope. The lack of interaction information makes it difficult for collaborative filtering to identify other users with similar interests to the new user, and therefore to recommend appropriate items for that user. In terms of recommendation based on graph neural networks, Ying et al. proposed PinSage, the first application of GNN in the field of commercial recommendation, which achieved good results. Wang et al. proposed KGCN to apply knowledge graphs to the recommendation field while combining graph neural networks to mine the relationship of commodities on the attributes of knowledge graphs, effectively capturing The inner connection of products is effectively captured. Subsequently, Wang et al. proposed KGAT to incorporate an attention-based aggregation approach to solve the problem of higher-order relationship inequality. Heterogeneous graph networks contain multiple types of nodes and edges due to their inclusion. It contains richer semantic information, and more general recommendation algorithms based on heterogeneous graphs are relatively less studied. Therefore, the purpose of this paper is to study graph

neural network-based recommendation algorithms, mainly considering the heterogeneity of the network structure and the interaction order information between nodes, and then build a new recommendation model to cope with the cold start problem and improve the recommendation effect.

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] M. B. Dias, D. Locher, M. Li, W. El-Deredy, and P. J. Lisboa, "The value of personalised recommender systems to e-business: a case study," in *Proceedings of the 2008 ACM conference on Recommender systems*, 2008, pp. 291–294.

[2] Y. Deldjoo, M. Schedl, B. Hidasi, Y. Wei, and X. He, "Multimedia recommender systems: Algorithms and challenges," in *Recommender systems handbook*. Springer, 2022, pp. 973–1014.

[3] B. Marlin and R. S. Zemel, "The multiple multiplicative factor model for collaborative filtering," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 73.

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

[5] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems." in *IJCAI*, vol. 17. Melbourne, Australia, 2017, pp. 3203–3209.

[6] 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.

[7] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE transactions on knowledge and data engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[8] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, and H.-P. Kriegel, "Probabilistic memory-based collaborative filtering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 1, pp. 56–69, 2004.

[9] 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.

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

[11] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

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

[13] 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.

[14] 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.

[15] 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.

[16] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[17] 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.

[18] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[19] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs," *IEEE transactions on signal processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

[20] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete signal processing on graphs: Sampling theory¡? pub _newline=""?" *IEEE transactions on signal processing*, vol. 63, no. 24, pp. 6510–6523, 2015.

[21] M. Welling and T. N. Kipf, "Semi-supervised classification with graph convolutional networks," in *J. International Conference on Learning Representations (ICLR 2017)*, 2016.

[22] A. Micheli, "Neural network for graphs: A contextual constructive approach," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009.

[23] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[24] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[25] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 1096–1103.

[26] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[27] Q. Wang, Y. Wei, J. Yin, J. Wu, X. Song, and L. Nie, "Dualgnn: Dual graph neural network for multimedia recommendation," *IEEE Transactions on Multimedia*, 2021.

[28] R. Chen, Q. Hua, Y.-S. Chang, B. Wang, L. Zhang, and X. Kong, "A survey of collaborative filtering-based recommender systems: From traditional methods to hybrid methods based on social networks," *IEEE Access*, vol. 6, pp. 64 301–64 320, 2018.

[29] 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.

[30] R. v. d. Berg, T. N. Kipf, and M. Welling, "Graph convolutional matrix completion," *arXiv preprint arXiv:1706.02263*, 2017.

[31] J. Zhang, X. Shi, S. Zhao, and I. King, "Star-gcn: Stacked and reconstructed graph convolutional networks for recommender systems," *arXiv preprint arXiv:1905.13129*, 2019.

[32] 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.