Heterogeneous Graph Neural Network for Multi-behavior Feature-Interaction Recommendation

Li Ma, Zheng Chen, Yingxun Fu, and Yang Li(✉)

North China University of Technology, Shijingshan District, Beijing, China
liyang17@mails.jlu.edu.cn

Abstract. Graph neural network has great advantage in learning vector representation of users and items for modern recommender systems. Modeling user-item interaction bipartite graph is helpful for learning the collaborative signals between users and items. However, this modeling scheme ignores the influence of the objectively existing attribute information of item itself, and cannot well explain why users focus on items.

A feature interaction-based graph convolutional collaborative filtering algorithm, Attention Interaction Graph Collaborative Filtering (ATGCF) is proposed to address the limitations of existing works in this paper. It can deeply explore user preferences for multi-feature items. Firstly, we inject user and multi-feature into a user-feature interaction layer composed of multi-head-attention and fully connected layers to capture the user potential preference for multi-feature. Then we build a user-features-item bipartite graph and design the corresponding graph aggregation layer to obtain the high-order connectivity between the three. Experiment results on three real-world datasets verify the effectiveness of our model in exploiting multi-feature data, and ATGCF algorithm outperforms the other baselines in terms of recall, precision and NDCG evaluation metrics. Moreover, further experiments demonstrate that our model can achieve better results with less data.

Keywords: Multi-head attention · Feature interaction · Collaborative filtering · Heterogeneous graph networks · Recommendation system

1 Introduction

Personalized recommender system has been widely used to deal with the problem of information overload [1]. Collaborative filtering (CF) [2], the most widely used recommendation model can mine user preference and estimate preference based on existing historical behavior data such as purchases and browsing. Traditional CF models [3] are designed to directly model the relationship between users and items, which can directly bring benefits to the platform, such as movie recommendations and music recommendations. However, in real-world applications, this may lead to serious data sparsity issue. For example, in movie software, it is often difficult to achieve better results by building a CF for recommendation only through user click behavior. In other words, recommender

E. Pimenidis et al. (Eds.): ICANN 2022, LNCS 13532, pp. 101–112, 2022.
https://doi.org/10.1007/978-3-031-15937-4_9

system should have the ability to utilize richer information, including interrelated movie features. Learning different feature combinations can help our model to more accurately predict the objects that users may interact with in the future.

Existing researches [4] has explored this task from two perspectives. The first category tries to mine information in the form of matrix factorization. [5] models the relationship between users and items directly through the historical interaction matrix and embedding the user vector and item vector through the dot product of the two vectors. Neural Collaborative Filtering (NCF) [6] replaces the interaction function of MF by introducing a deep learning method, so that the model can adaptively learn high-order vector representations of users and items. The second category considers modeling user historical behavior as a graph structure. [7] encodes the collaborative information between the user and the item, so that the model can learn the collaborative information hidden in the historical interaction among user and item to represent the behavioral similarity between users and items. [8] captures user preferences for each interacted item by adding node-level attention. However, these algorithms face the following two problems. First, only modeling user-item interactions will lead to very sparse data and seriously affect the recommendation effect. Especially for users with few interactive items, the noisy data may prevent the model from learning effective information from sparse historical behaviors. Second, only using user and item data will lose very rich attributes information. The items that the user has interacted with may have a certain similarity in features. For example, the user has watched more romantic anime movies in the past period of time, indicating that the user has a preference for romantic anime to a certain extent, so these multi-feature should be fully utilized.

To address them, we propose to construct a unified heterogeneous graph based on multi-features interaction. Firstly, we use users, items, features represented as nodes, and different types of behaviors represented as multiple types of edges of the graph to model user-feature and user-item-feature relationships, respectively. Secondly, we design a new user-item-features message passing graph recommendation model, Attention Interaction Graph Collaborative Filtering (ATGCF), which takes into account the historical interaction between users and items, also models the historical behavior response of user to the item feature preferences. To be more specific, we design a user and multi-feature interaction layer by utilizing user and features information to capture the preference information between user and features. Finally, we design user-item-features propagation layers, which helps to capture different CF semantics of item and multi-features similarity for user and enhance the learning for node embeddings. To summarize, our work makes the following contributions:

- We model the user-item-feature heterogeneous bipartite graph from the real scene and mine user preferences for multi-feature.
- We propose the Attention Interaction Graph Collaborative Filtering (ATGCF) model which can capture the internal associations between user and item features more accurately.
- We conduct experimental analysis on three public large-scale datasets. The performance of our proposed model is significantly improved compared to the other baselines in terms of accuracy, recall, and NDCG.

2 Methodology

2.1 Heterogeneous Bipartite Graph

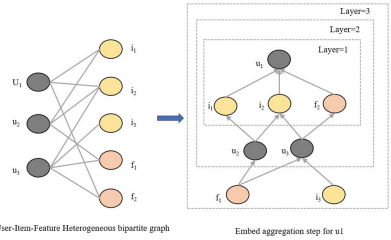

Fig. 1. Representation of the user-item-feature heterogeneous bipartite graph, and how users obtain information of all types of nodes as the network aggregates.

The heterogeneous bipartite graph shows the heterogeneity of the user next-hop node, and we describe the specific method of generating the graph in Fig. 1. First, we can obtain the user-item pair $\langle U, I \rangle$ through the historical interaction information. Meanwhile, we generate the item-feature pair $\langle I, F \rangle$ according to the attribute column of the item, and then the user-feature pair $\langle U, F \rangle$ can be obtained through the item as an intermediary. On the basis of obtaining the above information, the matrix R can be generated. Finally, the adjacency matrix A of our generated graph.

$$A = \begin{bmatrix} 0 & R \\ R^T & 0 \end{bmatrix} \tag{1}$$

The concatenation matrix representation of the adjacency matrix A obtained should satisfy $A = A^T$. That is to say, it satisfies $A = D^{\frac{1}{2}} \Lambda D^{-\frac{1}{2}}$, i.e. the graph volume operator in ATGCF satisfies the definition of Laplacian operator in GCN [9]. We refer to the mainstream recommendation model [10] to map users, items, features node into a vector. The final preference of each user for an item is predicted by finally learning embedding representation vector.

2.2 User-Features Interaction

In recommendation scenarios, such as movie recommendation and product recommendation. It is difficult for us to obtain the contextual information of the user, but the contextual information of the item can be obtained effectively. Our goal is to model the user's own preference for such items. First, we define the context of the item as where n represents the number of contextual features, and d represents the dimension of contextual features. However, in the absence of clear user features and item features, the user embedding will be inconsistent with the item embedding, which will make the model unable to train normally. Therefore, we obtain the user feature initialization by averaging

the sum of all the features of the items that the user has interacted with according to the historical interaction information between user and item. Expressed as:

$$F_u = \frac{1}{sum(R_{ui} \in E)} (\sum_{i \in R_{ui}} F_i) \tag{2}$$

where R_{ui} represents the interaction set between user and item history, $sum()$ represents the total number, and F_i represents the feature of item. Then, we define the feature embedding matrix $e_f \in R^{n \times d}$ to obtain the embedding representation of the feature. We map different features to the same low-dimensional space for convenience, and then combine different forms of features to represent the user's preference for different feature combinations and extract the semantic information of different feature combinations. We construct the user's preference for different types of items through the multi-head attention model [11]. Specifically, we use multi-head attention to analyze which items are more in line with the user preference. We map the embedded representations of users and features and user features to a low-dimensional space of the same dimension so that we can model higher-order interaction representations between users and different features. Next, we will elaborate on how we define it. First, we define the representation of different item feature combinations, the specific definitions are as follows:

$$\alpha_{i,m}^{h} = \frac{exp\left(\psi^{(h)}(F_u^i, F_u^m)\right)}{\sum_{l=1}^{M} exp\left(\psi^{(h)}(F_u^i, F_l^m)\right)} \tag{3}$$

where i and m represent the currently calculated user features, h represents the set number of heads, and M represents the total number of item features. $\psi^{(h)}(F_u^i, F_u^m) = <\mathrm{W}_q^{(h)}\mathrm{F}_u^i, \mathrm{W}_k^{(h)}\mathrm{F}_u^m>$ is the size of the correlation coefficient between current user feature i and user feature m. $W_{\mathrm{q}}^{(h)}$ and $W_k^{(h)}$ are weights. The final correlation coefficient between user feature i and user feature m on each head h is normalized by (3) and then expressed as $\alpha_{i,m}^{h}$. The feature-interaction is expressed as:

$$e_{ui}^{m} = \sum_{h=1}^{H} \sum_{k=1}^{M} \alpha_{F_u^m, F_u^i}^{(h)} (W_V^{(h)} e_{F_u^i}) \tag{4}$$

where $W_V^{(h)}$ is weight, e_{ui}^{m} represents the interactive embedding representation of user feature i and user feature m. Then we can define a user preference for different user feature interaction embedding. Its definition is shown as following:

$$\alpha_u^m = \sum_{h=1}^{H} \frac{exp\left(\psi^{(h)}\left(e_u, e_u^m\right)\right)}{\sum_{l=1}^{M} exp\left(\psi^{(h)}\left(e_u, e_l^m\right)\right)} \tag{5}$$

where $\psi^{(h)}\left(e_u, e_u^m\right)$ represents the user preference for each interaction feature, h represents the number of heads, and m represents the total number of feature combinations. The finally learned result represents the user preference for the current combined feature. It is expressed as:

$$e_{F_u} = ReLU\left(\sum_{h=1}^{h} \sum_{l=1}^{M} \alpha_u^{m(h)} \left(W_V^{(h)} e_u^{m(h)}\right)\right) \tag{6}$$

where $ReLU()$ represents the nonlinear activation function, $\alpha_u^{m(h)}$ and $e_u^{m(h)}$ represent the preference coefficient and preference vector on each head.

2.3 Graph Neural Network Aggregation Layer

The historical interaction information between users and items is a reliable source of user preference information, and we refer to the method of [6] to model the user's higher-order preference for items and item features by considering the features of the items. The specific graph convolution operation is shown as follows:

$$m_{u\leftarrow i}^{(k)}=\frac{1}{\sqrt{|N_u||N_i|}}(W_1^{(k)}e_i^{(k-1)}+W_2^{(k)}e_{F_i}^{(k-1)}+W_3^{(k)}(\left(e_i^{(k-1)}\oplus e_{F_i}^{(k-1)}\right)\odot e_u^{(k-1)})) \quad (7)$$

$$m_{i\leftarrow u}^{(k)}=\frac{1}{\sqrt{|N_u||N_i|}}(W_1^{(k)}e_u^{(k-1)}+W_2^{(k)}e_{F_u}^{(k-1)}+W_3^{(k)}(\left(e_i^{(k-1)}\oplus e_{F_i}^{(k-1)}\right)\odot e_u^{(k-1)})) \quad (8)$$

where e_{F_i}, e_{F_u} denote the item and feature embedding of the user, W_1, W_2, W_3 denote the coefficient matrix of the science department, e_i denotes the embedding representation of the item, and k denotes the current number of layers. Through the stacking of multi-layer networks, the model can learn the influence of multi-hop neighbors on current node. Meanwhile, the model updates the state representation of current node according to the state of multi-hop neighbors at last. After passing through the stacking of K layers, vector representation of users and items can be displayed as:

$$e_u^{(K)}=ReLU\left(m_{u\leftarrow u}^{(K)}+\sum_{i\in N_n}m_{u\leftarrow i}^{(K)}\right),\quad e_i^{(K)}=ReLU\left(m_{i\leftarrow i}^{(K)}+\sum_{i\in N_u}m_{i\leftarrow u}^{(K)}\right) \quad (9)$$

where the final learned user vector is denoted as e_u, and the item embedding is denoted as e_i. It should be noted that after combining item features and item features indirectly represented by item features, the dimensions of user embedding e_u and item embedding e_i are updated from the original $R^{n\times d}$ set to $R^{n\times 2d}$.

2.4 Prediction Layer

After the above propagation, we learn the user preference for each item. This is very meaningful. We can clearly understand why users choose this item and also learn that items with this attribute are of interest to users. Then, we express the user preference for items of each attribute by making points between the user embedded representation and the embedded representation of items with different attributes.

$$y_{ATGCF}(u,i,f)=e_u^T e_i \quad (10)$$

The user preference for different types of items can be expressed in a variety of ways. For example, calculating the distance between each embedding is used to represent the user's preference for different kinds of items. In this paper, we use the dot product [9] which was often used in existing researches.

2.5 Model Training

The overall model process is shown in Fig. 2. We use the BPR [12] loss to train the model. For each user, we calculate the preference for positive and negative samples, and then update the learnable parameters according to the difference between positive and negative samples. The overall loss is calculated as follows:

$$Loss = \sum_{(u,i,j)\in O} -ln\sigma\left(y_{ui} - y_{uj}\right) + \lambda\|\Theta\|^2 \tag{11}$$

where $O = \{(u,i,j)|(u,i) \in R^+, (u,j) \in R^-\}$ represents the paired training samples, and $\langle u,i\rangle$ represents the set of positive samples and $\langle u,j\rangle$ represents the set of negative samples. $\sigma()$ represents the activation function, and Θ represents the trainable parameters in the model and λ controls the L_2 regularization strength to prevent overfitting. The optimizer chooses Adam [13] in our proposed algorithm.

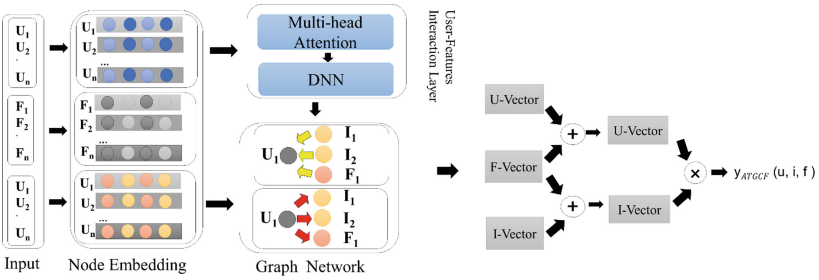

Fig. 2. An illustration of ATGCF model architecture

2.6 Complexity Analysis

In terms of complexity, the ATGCF model is simpler than the traditional CF model. Assuming our input matrix dimension is $R^{d\times d}$, the number of layers is L, and the number of nodes is n. Compared to MF which is the most concise embedding based on the recommendation model, our ATGCF only uses d_l^2 level parameters in the graph convolution layer. The overall complexity is $O\left(n^2d + dd_{l-1}\right)$. The additional cost of such model parameters is almost zero and negligible, considering that L is usually a number less than or equal to 3, and d_l is typically set as the embedding size, which is much smaller than the number of users and items.

3 Experiment

3.1 Dataset Description

To evaluate the effectiveness of ATGCF, we conduct experiments on three benchmark datasets: Amine, MovieLens-1m, and Amazon-book, which are publicly accessible and vary in terms of domain, size, and sparsity. We summarize the statistics of three datasets in Table 1.

Amine: This is a dataset from Kaggle website for anime recommendation. For each user we keep at least 20 more interactions.

MovieLens-1m: A public movie recommendation dataset for scientific research [14].

Amazon-Book: This is a widely used product recommendation dataset [15]. Similarly, it is also guaranteed that each user interacts with at least 20 items.

Table 1. Statistics of the datasets.

Dataset	#Users	#Items	#Interactions
Amine	15506	34325	2601998
MovieLens-1m	6040	5953	1000209
Amazon-Book	51639	84355	2648963

We consider each interaction directly observed in the training set to be recorded as a positive instance and randomly sample as a negative sample from the set of items that user has never interacted with.

3.2 Experimental Settings

Evaluation Metrics. For each method the output the user's preference scores over all the items. To evaluate the effectiveness of top-K recommendation and preference ranking, we adopt three widely-used evaluation protocols: recall@K, precision@K and ndcg@K. We evaluate the results of our model under different K values. We demonstrate the average metrics for all users in datasets.

To demonstrate the effectiveness, we compare our proposed NGCF with the following baselines:

- **ItemCF:** The algorithm learns item representations based on the historical interaction records of user items, considers that similar item vector representations are close, and finally recommends items based on similarity.
- **MF:** This is matrix factorization optimized by the Bayesian personalized ranking (BPR) loss, which exploits the user-item direct interactions only as the target value of interaction function.
- **NeuMF:** The method is a state-of-the-art neural CF model which uses multiple hidden layers above the element-wise and concatenation of user and item embeddings to capture their non-linear feature interactions.
- **NGCF:** NGCF is the state-of-the-art graph neural network model which has some special design to fit graph neural network into recommender system. Here only target behavior is used to build the user-item bipartite graph.
- **PUP** [8]**:** The state-of-the-art graph neural network model which has some special design to fit sensitivity of a user on item price. Effectively establish the potential relationship between user to commodity and commodity to price.

Parameters Settings. Our ATGCF is implemented in Pytorch. The embedding size is fixed to 64 for all models. We optimize all models with Adam optimizer, and the batch size is 4096. In terms of hyperparameters, we apply a grid search for hyperparameters: the learning rate is tuned amongst [0.0001, 0.0005, 0.001, 0.005], and for baseline models we randomly select 80% of historical interactions of each user to constitute the training set, and treat the remaining as the test set. When training ATGCF, we found that only 10% of the data is needed to achieve the best results. It demonstrates that our model has good performance in the face of small samples. L_2 normalization coefficient is tuned in [1e7, 1e6, 1e5, 1e4, 1e3].

3.3 Overall Performance

We compare the performance of our proposed algorithm ATGCF with all other baselines. The results on three datasets are reported on Table 2, Table 3 and Table 4. From the results, we have the following observations.

Model Effectiveness. Based on these tables, we find that our ATGCF overperforms all baselines substantially on all Recall@K, Precision@K and NDCG@K metrics. The average improvement of our model to the best baseline is 5.62%, 3.95% and 5.20% for Recall, Precision and NDCG on the Amine dataset; 3.60%,5.95% and 6.00% on MovieLens-1m dataset; 4.07%, 6.80% and 4.30% on Amazon-Book dataset, which justifies the effectiveness of our model.

Table 2. Comparisons on Amine and improvement comparing with the best baseline.

Method	Recall@20	Recall@30	Recall@40	Recall@50	Precisio@10	Precisio@20	Ndcg@20
ItemCF	0.0286	0.0304	0.0401	0.0412	0.0260	0.0201	0.0115
MF	0.0354	0.0446	0.0524	0.0644	0.0311	0.0231	0.0214
NCF	0.0402	0.0542	0.0644	0.0685	0.0342	0.0256	0.0256
NGCF	0.0434	**0.0580**	0.0669	0.0772	**0.0358**	0.0294	0.0294
PUP	**0.0482**	0.0574	**0.0684**	**0.0784**	0.0336	**0.0296**	**0.0304**
ATGCF	**0.0535**	**0.0616**	**0.0702**	**0.0806**	**0.0372**	**0.0308**	**0.0320**
Improve	**10.9%**	**6.2%**	**2.6%**	**2.8%**	**3.9%**	**4.0%**	**5.2%**

Table 3. Comparisons on MovieLens-1m and improvement comparing with the best baseline.

Method	Recall@20	Recall@30	Recall@40	Recall@50	Precisio@10	Precisio@20	Ndcg@20
ItemCF	0.0214	0.0238	0.0446	0.0512	0.0332	0.0327	0.0396
MF	0.0384	0.0425	0.0584	0.0748	0.0559	0.0445	0.0425
NCF	0.0401	0.0554	0.0644	0.0864	0.0532	0.0496	0.0489
NGCF	0.0443	0.0647	0.0839	0.1026	**0.0604**	0.0584	0.0504
PUP	**0.0455**	**0.0665**	**0.0856**	**0.1048**	0.0588	**0.0596**	**0.0548**
ATGCF	**0.0482**	**0.0689**	**0.0884**	**0.1070**	**0.0645**	**0.0627**	**0.0581**
Improve	**5.9%**	**3.6%**	**3.2%**	**2.0%**	**6.7%**	**5.2%**	**6.0%**

Table 4. Comparisons on Amazon-Book and improvement comparing with the best baseline.

Method	Recall@20	Recall@30	Recall@40	Recall@50	Precisio@10	Precisio@20	Ndcg@20
ItemCF	0.0184	0.0196	0.0312	0.0452	0.0186	0.0195	0.0356
MF	0.0250	0.0267	0.0524	0.0624	0.0254	0.0201	0.0518
NCF	0.0265	0.0288	0.0644	0.0665	0.0262	0.0225	0.0542
NGCF	0.0348	0.0387	0.0669	0.0731	**0.0328**	0.0257	**0.0630**
PUP	**0.0382**	**0.0415**	**0.0684**	**0.0754**	0.0316	**0.0285**	0.0624
ATGCF	**0.0408**	**0.0426**	**0.0713**	**0.0775**	**0.0358**	**0.0298**	**0.0651**
Improve	**6.8%**	**2.6%**	**4.2%**	**2.7%**	**9.1%**	**4.5%**	**4.3%**

Multi-features Models Work Well. In the comparison of MF, NCF, and NGCF, we can find that adding item features information into predicting (PUP and ATGCF) can improve the performance. And through the results, we find that ATGCF outperforms PUP on average by 10.9% on Recall@20, 4% on Precision@20, and 5.2% on NDCG@20, which also shows that multi-feature fusion will be better than single-feature fusion.

Our Model is the Best Model that can Mine Users' Potential Preferences for Items with Multi-features. Comparing with the graph neural network models that can extract user potential preferences such as NGCF and PUP, our ATGCF performs the best. That can be explained in two folds. In terms of item preference, neither NGCF nor PUP has uncovered the user deep preference for interacted items, which may lead to the loss of some information in original data. In terms of feature processing, PUP only considers the impact of a single feature (price) on user selection and does not comprehensively consider the situation of multi-features. Fortunately, our ATGCF fully considers the influence of multiple features on user selection, and can dig out the user preference for different mixed features.

3.4 Model Analysis

As the feature interaction propagation layer plays a pivotal role in ATGCF, we investigate its impact on the performance. We first explore the impact of different feature embedding methods on the model, and then study the performance of ATGCF with different proportions of training data.

ATGCF-F. Learning by directly adding the features of items and then integrating them into the network. The method neither understands user preferences for different features nor the impact of different feature combinations.

ATGCF-IF. Learning the user's preference for each item feature by using the self-attention mechanism. On the basis of considering features, the user's preferences for different features are considered.

ATGCF. Modeling user preferences for different interaction features based on consideration of different feature preferences. The user preference for different feature combination items is learned through the multi-head attention mechanism plus MLP.

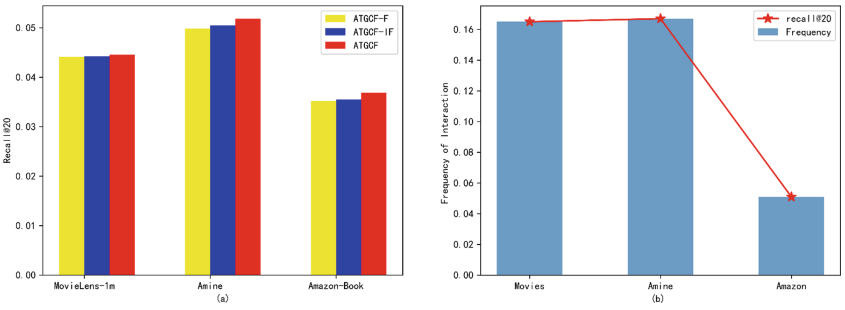

Fig. 3. (a) The influence of different feature extraction components on ATGCF, (b) The sensitivity analysis of ATGCF to user's feature preference under different interaction frequencies.

In order to analyze the necessity of each feature selection method, we use the three methods (ATGCF-F, ATGCF-IF and ATGCF) to extract the user preferences for different item features. The results are shown in Fig. 3(a), the effect of learning interaction features from the results is significantly better than that of using the attention mechanism to learn the users preference for item features. Meanwhile, using the attention mechanism to learn the user's preference for different items also achieve better performance than simply adding item features. Therefore, the mechanism we design to learn user preferences for items with different combinations of features is necessary and can obtain better performance.

Meanwhile, we analyze the sensitivity of ATGCF to features. We use the total number of interactions in the dataset divided by the number of users in the dataset (emphasizing the role of features) define the user interaction frequency, and obtain the interaction frequency size of MovieLens-1m 165, Amine 167 and Amazon-Book 51, separately. As

long as the interaction frequency is high, it means that we can extract more user's selection history of items. Moreover, we can more accurately extract the user's preference for item features. The experimental results are shown in Fig. 3(b). The higher the interaction frequency of the dataset, the better the effect of the ATGCF model. It also demonstrates the sensitivity of our model to item features and the rationality of our model design.

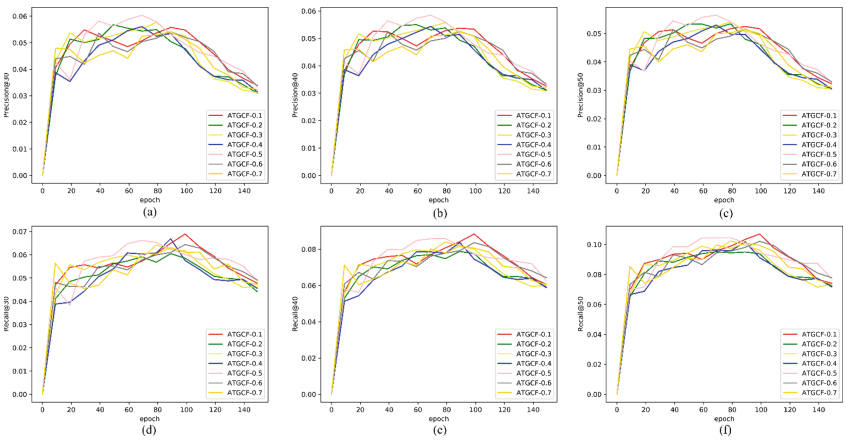

Fig. 4. Analysis of the recall and accuracy results of ATGCF with different training ratios in MovieLens-1m.

We evaluate the performance of our proposed algorithm with different training ratios and find that ATGCF requires less training data and achieve better performance. As shown in Fig. 4, the ordinate represents the performance value achieved by each method. The decimal at the end of each method in the lower right corner represents the ratio of the training samples to the total dataset. We find that while the ratio of 0.5 performs the best, ATGCF also achieves relatively good performance with a ratio of 0.1. The baseline model achieves the best effectiveness at a ratio of 0.7–0.8 [8]. The results indicates that even with little user interaction data, better recommendations can be made to users on the premise of understanding user preferences for item features. However, as the number of training increases, overfitting will occur, and we guess that the model fits unrepresentative features due to too many training times.

4 Conclusion

In this work, we study the feature interaction graph neural network recommendation model, and propose a graph neural network model ATGCF. We inject user and multi-features into a user-features interaction layer composed of multi-head-attention and fully connected layers to capture the user potential preference for multi-features. Then we build a user-features-item bipartite graph and design the corresponding graph aggregation layer to obtain the high-order connectivity among user, features and items. Ultimately, experiments on three public datasets demonstrate that our model outperforms the best baseline model in 4.43%, 5.56% and 5.16% for Recall, Precision and NDCG.

Acknowledgement. This work was supported by National Key R&D Program of China (2018YFB1800302), Natural Science Foundation of China (62001007), Beijing Natural Science Foundation (KZ201810009011, 4202020, L192021, 4212018), Re-search Start-up Fund of North China University of Technology and Jilin Province Science and Technology Development Plan Project (20190201180JC, 20200401076GX).

References

1. Wu, L., He, X., Wang, X., et al.: A survey on accuracy-oriented neural recommendation: from collaborative filtering to information-rich recommendation. IEEE Trans. Knowl. Data Eng. (2022)
2. Chen, C.M., Wang, C.J., Tsai, M.F., Yang, Y.H.: Collaborative similarity embedding for recommender systems. In: Proceedings of the 28th International Conference on World Wide Web, pp. 2637–2643 (2019)
3. Lin, Z., et al.: A structured self-attentive sentence embedding. arXiv preprint arXiv:1703.03130 (2017)
4. Jin, B., Gao, C., He, X., Jin, D., Li, Y.: Multi-behavior recommendation with graph convolutional networks. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 659–668 (2020)
5. Qiu, J., et al.: GCC: graph contrastive coding for graph neural network pre-training. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1150–1160 (2020)
6. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the 26th International Conference on World Wide Web, pp. 173–182 (2017)
7. He, R., McAuley, J.: Ups and downs: modeling the visual evolution of fashion trends with one-class collaborative filtering. In: Proceedings of the 25th International Conference on World Wide Web, pp. 507–517 (2016)
8. Zheng, Y., Gao, C., He, X., Li, Y., Jin, D.: Price-aware recommendation with graph convolutional networks. In: Proceedings of IEEE 36th International Conference on Data Engineering, pp. 133–144 (2020)
9. Berg, R., Kipf, T.N., Welling, M.: Graph convolutional matrix completion. arXiv preprint arXiv:1706.02263 (2017)
10. Cao, Y., Wang, X., He, X., Hu, Z., Chua, T.S.: Unifying knowledge graph learning and recommendation: towards a better understanding of user preferences. In: Proceedings of the 28th International Conference on World Wide Web, pp. 151–161 (2019)
11. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems 30 (2017)
12. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. arXiv preprint arXiv:1205.2618 (2012)
13. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
14. Harper, F.M., Konstan, J.A.: The movielens datasets: history and context. ACM Trans. Interact. Intell. Syst. **5**(4), 1–19 (2015)
15. Ricci, F., Rokach, L., Shapira, B.: Introduction to recommender systems handbook. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 1–35. Springer, Boston, MA (2011). https://doi.org/10.1007/978-0-387-85820-3_1