

A Cross-Domain Recommendation Algorithm Based On Graph Optimization

Zheng Fan

Electronic Engineering College
Heilongjiang University
No.74, Xuefu Road, Harbin, China
fanzheng07@outlook.com

Ying-Li Wang

Electronic Engineering College
Heilongjiang University
No.74, Xuefu Road, Harbin, China
wangyingli@hlju.edu.cn

Qi-tao Ma

China Mobile Group Shanghai Co., Ltd.
No.200, Changshou Road, Putuo District, Shanghai, China
jack_coldsweat@163.com

Hai-Xia Du

Electronic Engineering College
Heilongjiang University
No.74, Xuefu Road, Harbin, China
hxd1410501@163.com

Hong-Bin Ma*

Electronic Engineering College
Heilongjiang University
No.74, Xuefu Road, Harbin, China
mahongbin@hlju.edu.cn

*Corresponding author: Hong-Bin Ma

Received December 27, 2022, revised January 18, 2023, accepted March 1, 2023.

ABSTRACT. *In the recommendation system, the accuracy of the recommendation results may be reduced because the user has less historical operation data, which is the challenge problem of data sparsity. The cross-domain recommendation system can effectively solve the problem of data sparsity by supplying the target domain with data from other domains. However, the issue is that there is noise in the information about how users and items interact, such as user misoperations, etc., and these noises will affect the recommendation results. Therefore, if the impact of these noises can be effectively reduced, the user interactive information can be more fully utilized. In this study, we present a novel algorithm for cross-domain recommendation based on graph optimization. The algorithm optimizes the graph, deletes nodes with a high proportion of noise and less effective feature information, thereby reducing the impact of noise on training. In addition, the algorithm adds a feature transfer module to further increase the ability of the algorithm to capture user preferences. Experiments on two pairs of datasets prove that our algorithm is more effective than several current advanced algorithms.*

Keywords: Recommendation system, Data sparsity, Graph neural network, Graph optimization

1. **Introduction.** In the recent, the fast-growing e-commerce industry has attracted a large number of consumers with its advantages of convenience, speed and time saving. On the e-commerce website, hundreds of millions of new goods, texts, and videos are continually being released. Recommendation systems now play a crucial part in various applications, such as social platforms, news platforms, and short video platforms. E-commerce platforms introduce recommendation systems at the backend system to predict the potential relationship between users and products, so as to realize the personalized product recommendation process of users [1–5]. Therefore, a large number of user interaction records and historical click information are necessary to fully exploit the high performance of the recommendation system. This is because the historical interaction information of users reflects their interest direction, which can be used to predict what a user is likely to purchase in the near future [6,7]. However, in reality, we often encounter the problem of few user interaction records, which is not enough to forecast the interests of users well. This is the issue of data sparsity [8–11].

Deep learning algorithms are generally successful in processing Euclidean spatial data [12–14]. But due to the sparsity of data information, as well as the diversification, complexity, and disorder of data forms, traditional deep learning recommendation algorithms and collaborative filtering recommendation algorithms cannot solve the problem well [15–19]. Graph neural network can topologically represent information in non-Euclidean space, and is often used to process complex graph data. In addition, as users use more platforms, use the same account to browse different categories of products, and begin to interact in multiple fields(e.g., music, TV shows, books and movies), so that the issue of sparse data in the target field can be addressed using data gathered from other fields. Cross-domain recommendation based on graph neural networks utilize these principles to help achieve higher-performance personalized recommendations.

In order to enhance the efficacy of cross-domain recommendations in the target domain, contemporary cross-domain recommendation systems make greater use of interactive information in the source domain. Nonetheless, the problem is that there is noise in the interactive information, such as user misoperations, etc. Since these noises participate in the training of the model, they will greatly bias the output of the recommendation system. Therefore, if the weight of these noises can be reduced in model training, and the importance of other data that can represent user preferences can be strengthened, the recommended accuracy of the recommendation system could be greatly boosted [20].

In this study, we present a novel algorithm based on graph optimization. This method optimizes graphs, eliminates nodes that are not conducive to training, and impacts of redundant data are diminished on recommendation results. In addition, feature transfer is added to further improve the algorithm performance. Complete construction of the method is shown in Figure 1. By applying the novel cross-domain recommendation algorithm in this study, the influence of noise in the source domain can be minimized, and the user interactive information can be utilized more adequately. First, we use the users' purchase history data in the source and target domain to initially construct the user-item graph representation through the LightGCN [21] method, canonical correlation analysis (CCA) is then used to project the source and target domain representations of the users and items embedding features into the latent space. We compare the local densities of items with the local densities of adjacent items. Then, an anomaly detection method, the local outlier factor (LOF) method, is used to further identify outliers of the graph. Finally, we add the feature transfer module in the BiTGCF [22] method to better capture user preferences and further increase the recommended accuracy of the algorithm. The novel graph optimization method presented in this study can prominently reduce the adverse impact of noise. In the third part of the paper, the effectiveness of our method is verified through comparative experiments.

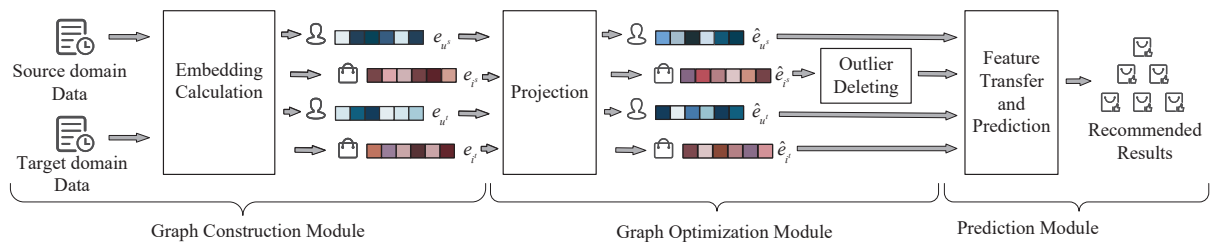


FIGURE 1. Complete construction of the method

2. Presented Model. In this section, we present the novel algorithm for cross-domain recommendation in detail. In Section 2.1, we first construct a preliminary graph to represent user and item embedding features. Then project the graph representations to the latent space. In Section 2.2, the graph optimization module is introduced, mainly by deleting the source domain nodes with little correlation in the latent space, which is the core innovation content in the article. Section 2.3 briefly describes the feature transfer and recommendation network applied in this paper, which is the current advanced recommendation algorithm based on bi-directional transfer graph collaborative filtering networks (BiTGCF). Section 2.4 introduces the final output project recommendation module.

2.1. Initialization of embedding features. In this module, we first perform a preliminary construction and representation of graphs for each domain. We use LightGCN method to initially construct the graph, which simplifies the feature propagation module but is very efficient, the network model architecture of LightGCN method is shown in Figure 2.

In the field of graph research, we often make the following definitions, user $u \in \{1, 2, \dots, U\}$, user's embedding features is represented by $e_u \in \mathbb{R}^d$, item's embedding features is represented by $e_i \in \mathbb{R}^d$, and d represents the size of embedding features. Then we use the LightGCN, which is currently an advanced and excellent graph neural network,

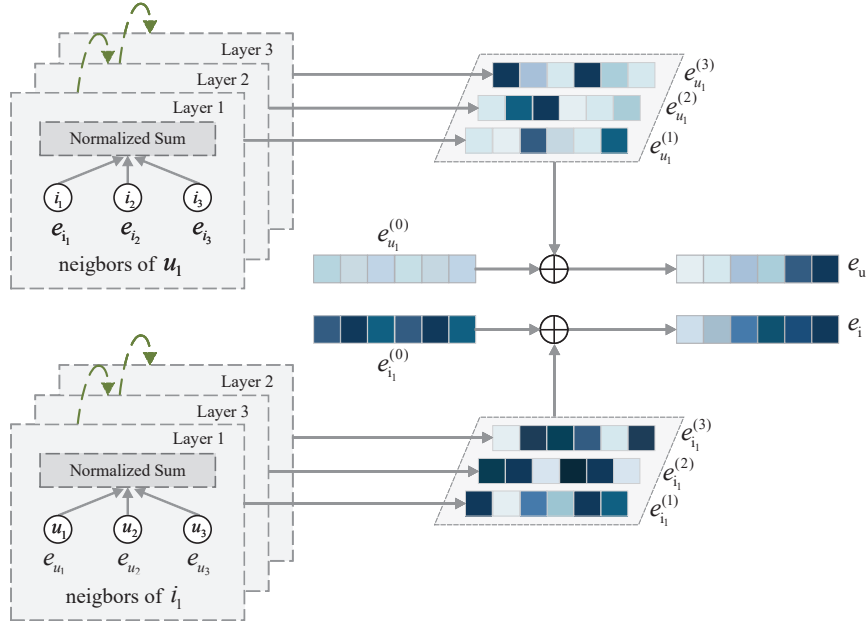


FIGURE 2. Illustration of LightGCN model architecture

to initially construct graph embedding features. We represent the graph by fusing the feature relationship of users and items, and add the embedded features of n -th neighbors to the target node weightedly. The nodes of this graph carry the feature information of nearby nodes, thus the model can strengthen the feature representation effect of the graph. The feature propagation layer we use is as follows:

$$e_u^{(n+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u| |\mathcal{N}_i|}} e_i^{(n)}, \quad (1)$$

$$e_i^{(n+1)} = \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i| |\mathcal{N}_u|}} e_u^{(n)}, \quad (2)$$

where n represents the number of graph convolutional network layers. $e_i^{(n)}$ and $e_u^{(n)}$ represent the embedding features of user u and item i in the n -th propagation layer respectively. The embedding features of user u and item i are obtained by the normalized sum of the embedding features of their neighbors. The symmetric normalization term $\frac{1}{\sqrt{|\mathcal{N}_i| |\mathcal{N}_u|}}$ comes from the standard graph convolutional neural network GCN, which prevents the graph embedding representation from growing larger and larger as the graph convolution operation proceeds. The graph convolution operation here only aggregates neighbor node information without self-connection. Then the embedding features calculated by each layer are linearly added to the target node:

$$e_u = \sum_{n=0}^N \alpha_n e_u^{(n)}, \quad (3)$$

$$e_i = \sum_{n=0}^N \alpha_n e_i^{(n)}, \quad (4)$$

α_n represents the weight of the n -th layer of embedding features in the action of constructing the final embedding feature representation, where it is a hyperparameter. Here we set α_n to $\frac{1}{n+1}$ to keep the algorithm with a good performance. In the action of our

initial construction of the graph representation, the only trainable parameter matrix is the embedded representation matrix $E^{(0)} \in \mathbb{R}^{(U+I) \times d}$. Here we use Bayesian Personalized Ranking (BPR) loss that encourage the prediction of liked samples to be higher than its disliked counterparts:

$$L_{\text{BPR}} = - \sum_{u=1}^U \sum_{i_l \in \mathcal{N}_u} \sum_{i_d \notin \mathcal{N}_u} \ln \sigma(y_{(u,i_l)} - y_{(u,i_d)}) + \varepsilon \|E^{(0)}\|_F^2, \quad (5)$$

where ε represents the hyperparameter of the L_2 norm, and $\sigma(\cdot)$ is the sigmoid function, a nonlinear activation function. We judge whether i_l and i_d are user u 's preference direction by estimating $y_{(u,i_l)}$ and $y_{(u,i_d)}$. We compute $y_{(u,i_l)}$ and $y_{(u,i_d)}$ by embedding feature representations:

$$y_{(u,i_l)} = e_u^\top e_{i_l}, \quad (6)$$

$$y_{(u,i_d)} = e_u^\top e_{i_d}, \quad (7)$$

where i_d represents the items that user u does not like, and i_l represents the items that user u likes. Then, we construct embedding feature representation by minimizing the loss function L_{BPR} . From this, we can obtain embedding features with high feature representations in each domain.

2.2. Graph optimization in latent space. In Section 2.1, we used the method in LightGCN to construct a preliminary graph of user u and item i in the source domain and target domain. The graph representation contains information about its own nodes and surrounding neighbors. Next, we project these graph feature representations into the latent space. We consider two notations D_s and D_t representing source domain and target domain respectively. According to the idea of the CCA algorithm, we need to determine the projection vectors $a \in \mathbb{R}^{d_s \times d_{cca}}$ and $b \in \mathbb{R}^{d_t \times d_{cca}}$ to compare the correlation between the embedding features $E_{u^s} = [e_{1^s}, e_{2^s}, \dots, e_{U^s}] \in \mathbb{R}^{d_s \times U}$ and $E_{u^t} = [e_{1^t}, e_{2^t}, \dots, e_{U^t}] \in \mathbb{R}^{d_t \times U}$ of users in D_s and D_t . We first need to maximize the correlation coefficient of E_{u^s} and E_{u^t} :

$$\underbrace{\arg \max}_{a,b} \frac{a^\top S_{E_{u^s} E_{u^t}} b}{\sqrt{a^\top S_{E_{u^s} E_{u^s}} a} \sqrt{b^\top S_{E_{u^t} E_{u^t}} b}}, \quad (8)$$

where $S_{E_{u^s} E_{u^t}}$ represents the covariance $\text{cov}(E_{u^s}, E_{u^t})$ of E_{u^s} and E_{u^t} . We need to maximize the optimized coherence coefficient to obtain the corresponding projection vectors a , b , and the eigenvalue decomposition method is applied here. After we determine the values of the projection vectors a and b , we can acquire the projected feature representations $\hat{e}_{i^s} \in \mathbb{R}^{d_{cca}}$ and $\hat{e}_{i^t} \in \mathbb{R}^{d_{cca}}$ of item i in D_s and D_t :

$$\hat{e}_{i^s} = a^\top e_{i^s}, \quad (9)$$

$$\hat{e}_{i^t} = b^\top e_{i^t}. \quad (10)$$

Next we perform elimination in the latent space on nodes considered to be outliers. Nodes in D_s are first compared, and nodes that are closer in the latent space are more correlated, whereas nodes that are farther away are less correlated. So, if a node in the latent space is far away from its neighbor nodes, then this node is considered as an outlier that should be eliminated. The reachable distance between i^s and o is defined as the larger value of the k -th distance of i^s and the actual distance between i^s and o :

$$\text{reach-dist}_k(i^s, o) = \max(k\text{-dist}(i^s), d(i^s, o)). \quad (11)$$

Local reachability density formula for an item in D_s is defined:

$$\text{lrd}(i^s) = 1 / \left(\frac{\sum_{o \in N_k(i^s)} \text{reach-dist}_k(i^s, o)}{|N_k(i^s)|} \right). \quad (12)$$

The formula represents the reciprocal of the average reachable distance of all points in the k -th order neighborhood of the i^s node to i^s . The higher the value of the density $\text{lrd}(i^s)$, we think that i^s is more likely to be a cluster with the surrounding neighborhood points, the lower the value of the density $\text{lrd}(i^s)$, the more likely i^s is not a cluster with the surrounding neighborhood points, and the more likely i^s is an outlier. Next, we use the local outlier factor (LOF) to further determine whether i^s is an outlier:

$$\text{lof}(i^s) = \frac{\sum_{o \in N_k(i^s)} \frac{\text{lrd}(o)}{\text{lrd}(i^s)}}{|N_k(i^s)|}. \quad (13)$$

The value of $\text{lof}(i^s)$ represents the average of the ratio of the local reachability density of the neighborhood point $N_k(i^s)$ of the item node i^s to the local reachability density of the point i^s . The closer the value of $\text{lof}(i^s)$ is to 1, it indicates that $\text{lrd}(i^s)$ is close to the local reachability density of its neighborhood nodes. This indicates that node i^s and its neighbors belong to the same cluster. The value of $\text{lof}(i^s)$ is less than 1, indicating that $\text{lrd}(i^s)$ is higher, i^s is a dense point, and the greater the LOF is greater than 1, it indicates that $\text{lrd}(i^s)$ is smaller than the density of neighboring points. At this point we regard i^s as an outlier and delete it as noise. The graph-optimized embedding features $\hat{e}'_{i^s} \in \mathbb{R}^{d_{cca}}$ can achieve better feature representation, where $i^s \in \{1, 2, \dots, \hat{I}\}$, \hat{I} is the number of items in D_s after graph optimization. At this stage, graph optimization module based on outlier detection has been completed. The implementation steps of the graph optimization module is shown in Algorithm 1.

Algorithm 1 Graph optimization module based on outlier detection

Input: All samples of users in the source domain U ; all samples of items in the source domain I ; the interactive information between users and items $R: U \times I$

Output: Embedding features representation of items in D_s after graph optimization \hat{e}'_{i^s} .

- 1: Initially construct the graph representation G or the subgraph $G_{u,i}$;
 - 2: Iteratively propagate the information of neighbors and update the node embedding;
 - 3: **for** n in $[1, \dots, N]$ **do** (Take the item node as an example)
 - 4: $e_s^{i^{(l)}} = \text{Aggregator}(e_{i^s}^{(n)}, e_{i^s}^{(n)})$
 - 5: $e_s^{i^{(l+1)}} = \text{Updater}(e_s^{i^{(l)}}, n_s^{i^{(l)}})$
 - 6: **end for**
 - 7: // Get the overall item/user representation $E_{i^s} = [e_{1^s}, e_{2^s}, \dots, e_{I^s}]$, $E_{u^s} = [e_{1^s}, e_{2^s}, \dots, e_{U^s}]$
 - 8: $\hat{e}'_{i^s} \leftarrow \Psi_s^\top e_{i^s}$
 - 9: **for** i in $[1, \dots, I]$ **do**
 - 10: **if** $\text{lof}(i^s) > 1$ **then**
 - 11: $\hat{e}'_{i^s} = \text{Updater}(\hat{e}_{i^s})$
 - 12: $i^s = i^s - -$
 - 13: **end if**
 - 14: **end for**
 - 15: **Return:** \hat{e}'_{i^s}
-

2.3. Feature transfer and training strategies. In this section, we apply the BiTGCF method for feature transfer and item recommendation in D_t . We feed the user and item embedding features representations $\hat{e}_{u^s}^{(n)}$, $\hat{e}_{u^t}^{(n)}$, $\hat{e}_{i^s}^{(n)}$, $\hat{e}_{i^t}^{(n)}$ in D_s and D_t into an n -layer graph convolution network. The feature transfer module includes two modules of feature propagation and feature transfer, so as to update the embedding features. The formulas to realize feature propagation and feature transfer are as follows:

$$\begin{aligned}\hat{e}_{u^s}^{(n+1)} &= f_{\text{T}}^s \left(f_{\text{P}}^s(\hat{e}_{u^s}^{(n)}), f_{\text{P}}^t(\hat{e}_{u^t}^{(n)}) \right), \\ \hat{e}_{u^t}^{(n+1)} &= f_{\text{T}}^t \left(f_{\text{P}}^s(\hat{e}_{u^s}^{(n)}), f_{\text{P}}^t(\hat{e}_{u^t}^{(n)}) \right), \\ \hat{e}'_{i^s}{}^{(n+1)} &= f_{\text{P}}^s(\hat{e}'_{i^s}{}^{(n)}), \\ \hat{e}_{i^t}{}^{(n+1)} &= f_{\text{P}}^t(\hat{e}_{i^t}{}^{(n)}),\end{aligned}\tag{14}$$

where $\hat{e}_{u^s}^{(n)}$ and $\hat{e}'_{i^s}{}^{(n)}$ represent the embedding features of user u and item i in D_s after n -layer feature propagation. $\hat{e}_{u^t}^{(n)}$ and $\hat{e}_{i^t}{}^{(n)}$ represent the embedding features of user u and item i in D_t after n -layer feature propagation. $f_{\text{P}}^s(\cdot)$ and $f_{\text{P}}^t(\cdot)$ denote feature propagation functions in D_s and D_t respectively. $f_{\text{T}}^s(\cdot)$ and $f_{\text{T}}^t(\cdot)$ denote the feature transfer functions in D_s and D_t respectively. Connection structure for feature transfer module is shown in Figure 3. i_1^s obtains part of the feature information from its first-order neighbor u_1^s to update its embedding features representation. Similarly, u_1^t obtains part of the feature information from its first-order neighbor i_1^t to update its own embedding features representation. Here we can get the $i_1^s \rightarrow u_1^s \rightleftharpoons u_1^t \rightarrow i_1^t$ (or $i_1^s \leftarrow u_1^s \rightleftharpoons u_1^t \leftarrow i_1^t$) route to realize feature propagation and feature transfer between nodes.

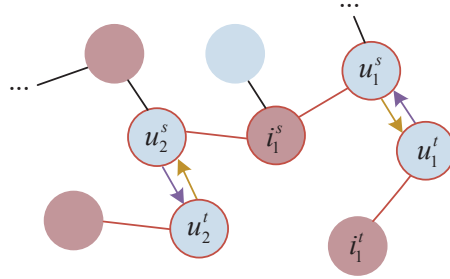


FIGURE 3. Connection structure for feature transfer module

After n layers of graph convolutional network, we obtain multiple refined embedding vectors of users and items. We bring more robust embeddings for users and items through the concatenation of multiple feature vectors. We obtain the final embedding vectors of the users and items in D_s and D_t by the following formulas:

$$\begin{aligned}\hat{e}_{u^s} &= \hat{e}_{u^s}^{(0)} \parallel \cdots \parallel \hat{e}_{u^s}^{(n)}, \\ \hat{e}_{u^t} &= \hat{e}_{u^t}^{(0)} \parallel \cdots \parallel \hat{e}_{u^t}^{(n)}, \\ \hat{e}'_{i^s} &= \hat{e}'_{i^s}{}^{(0)} \parallel \cdots \parallel \hat{e}'_{i^s}{}^{(n)}, \\ \hat{e}_{i^t} &= \hat{e}_{i^t}{}^{(0)} \parallel \cdots \parallel \hat{e}_{i^t}{}^{(n)},\end{aligned}\tag{15}$$

where \parallel denotes a concatenation operation. Finally, we use the dot product to get the predicted value of the interactive information between the users and the input items:

$$\hat{r}_{ui}^s = \hat{y}(u^s, i^s) = \sigma(\hat{e}_{u^s}^\top \hat{e}'_{i^s}),\tag{16}$$

$$\hat{r}_{ui}^t = \hat{y}(u^t, i^t) = \sigma(\hat{e}_{u^t}^\top \hat{e}_{i^t}), \quad (17)$$

where $\sigma(\cdot)$ is the sigmoid activation function that maps the actual value to the probability value, and \hat{r}_{ui} is the outcome prediction. We apply the binary cross-entropy function commonly used in recommendation systems as the loss function:

$$\mathcal{L}(\hat{r}_{ui}) = - \sum_{(u,i) \in R^+ \cup R^-} r_{ui} \log \hat{r}_{ui} + (1 - r_{ui} \log(1 - \hat{r}_{ui})) + \lambda \|\Theta\|_2^2. \quad (18)$$

Among them, R^+ is the historical data of user-item interaction known to exist in the dataset, and R^- is a random sample set without user-item interaction. λ is the parameter of L_2 regularization. Θ is all trainable graph representation matrices as input. In addition, we use mini-batch Adam [23] to speed up the process of iterating the model and updating the parameter matrix. Finally, to boost the recommended accuracy of the algorithm in D_s and D_t simultaneously, we apply the joint loss function, which is defined:

$$\mathcal{L}_{\text{join}} = \min_{f_{\text{T}}^s, f_{\text{P}}^s, f_{\text{T}}^t, f_{\text{P}}^t} \mathcal{L}(\hat{r}_{ui}^t, r_{ui}^t) + \mathcal{L}(\hat{r}_{ui}^s, r_{ui}^s). \quad (19)$$

2.4. Item recommendation. We determine item recommendation results using the BiTGCF model trained. We apply the result value \hat{r}_{ui}^t as the possibility of interactive information between user u and item i , and apply this as the ranking score in the recommendation system to rank the input items. The formula is defined as follows:

$$\hat{r}_{ui}^t = \sigma(\hat{e}_{u^t}^\top \hat{e}_{i^t}), \quad (20)$$

where item i with higher preference gets higher result value \hat{r}_{ui}^t for user u . Then, we rank the user's preference items and output the results in order.

3. Experiment. In this subsection, we first provide a detailed explanation of the datasets utilised in the performance testing, performance verification protocol, and custom settings in Section 3.1. We then analyze the performance of methods in Section 3.2. Finally, experiments are designed to analyze the relationship between the different values of the parameter k and the recommended accuracy of the algorithm in Section 3.3.

3.1. Experiment setting. A real dataset, the Amazon-5cores dataset [24], is employed in our experiments, which is collected in Amazon e-commerce platform. Each user and item in this dataset has at least five evaluation records, from which we select two pairs of fields for experimentation. The first pair of datasets is "Books" and "Movies and TV". The second pair of datasets is "CDs and Vinyl" and "Digital Music". We need to filter out overlapping users in each pair of domains, and the details of the filtered datasets are listed in Table 1.

We use the leave-one-out method commonly used to train and test the model of recommendation algorithms. We set up the test dataset to randomly pick a sample from each user's interactive information data, and set up the training dataset to the remaining interactive information data specifically. Then 99 items are chosen at random as negative samples from the items that user unpurchased. We use Hit Ratio (HR), Normalized Discounted Cumulative Gain (NDCG), and Mean Reciprocal Rank (MRR) in our experiments, which are commonly used as performance verification indicators in recommendation algorithms. For each measure, we weaned the ranking list to 10. We also visualize the output results of our method and the baseline method LightGCN, which can intuitively compare the relationship between the recommendation results of the two methods and the training input items in source domain. In the comparison experiment, the parameter k of the algorithm in this study is set to 5, which is the parameter value for the best

performance of the algorithm. The experiment of the relationship between the specific parameter value k and the algorithm is in Section 3.3.

TABLE 1. Details of the two couple experimental datasets

| Domain | #Users | #Items | #Interactions | Density |
|---------------|--------|---------|---------------|---------|
| CDs and Vinyl | 5,328 | 55,682 | 375,823 | 0.126% |
| Digital Music | 5,328 | 3,543 | 63,051 | 0.334% |
| Books | 37,376 | 269,283 | 1,253,568 | 0.012% |
| Movies and TV | 37,376 | 49,267 | 791,072 | 0.043% |

We use the algorithm Our Method (OM) presented in this study to compare with different recommendation algorithms, LightGCN [21], an effective method to remove nonlinear activation and simplify feature propagation; CDFM [25], an algorithm that uses factorization machines to generate recommendation results based on source domain context; CoNet [26], a cross-domain deep model that passes messages across domains across underlying networks. PPGN [16], a cross-domain recommendation system based on user preference propagation; BiTGCF [22], a bidirectional transitive graph collaborative filtering network; CATN [27], an aspect transfer network with an attention module. In addition, a performance comparison was made with the method Our Method without graph optimization (OM-W), which does not include the graph optimization module, to evaluate the graph optimization module’s performance in boosting the algorithm’s predictive accuracy.

3.2. Comparison of Experimental Results. The full results of our experiments on two pairs of datasets are shown in Table 2, the evaluation indicators are HR@10, MRR@10 and NDCG@10. The analysis of the results in the table shows that OM performs better than the majority of the baseline methods, which shows that OM based on the graph optimization has higher performance. However, in the domains of "Books" and "Movies and TV", the value of OM is not as high as the value of CATN for the indicator of NDCG@10. After analysis, the reason is probably that in the domains of "Books" and "Movies and TV" with relatively low density, CATN method of extracting multiple aspects and learn aspect correlations can better capture user’s preferences than OM. Therefore, the NDCG@10 value of CATN in the experimental results is higher than that of OM, but CATN is not as good as OM in other evaluation indicators. In addition, by comparing the experimental results of the two methods of OM-W and OM, we can clearly see that all the evaluation indicators of OM in the two pairs of data sets are better than OM-W, which shows that the presented graph optimization module in this study can significantly enhance the predictive accuracy of recommendation algorithms.

In order to show that the recommended items of OM are strongly correlated with the source domain items input during training, we compare OM with one of the baseline methods LightGCN. The domain selected in this experiment are "Books" and "Movies and TV". The visualization output of the two algorithms are shown in Figure 4. The input items of the experiment are nine books, all of which are about mystery. The five films output by OM are all about action and mystery, which can match the genre of the input projects. In addition, the content of (b) and (h) in the input book is about personal fate and life, and the output result of OM, (s) is also a movie about personal fate. In the input item, (e) is a mystery literary work including love and family, in the output item of OM, (p) is a mystery movie about love, and (q) is a mystery movie including family and love. In comparison, the output of LightGCN is also five action and mystery movies, but

TABLE 2. Details of the experimental results

| Metrics | Datesets | LightGCN | CDFM | CoNet | PPGN | BiTGCF | CATN | OM-W | OM |
|---------|---------------|----------|--------|--------|--------|--------|---------------|--------|---------------|
| HR@10 | CDs and Vinyl | 0.0100 | 0.6935 | 0.7539 | 0.7842 | 0.7815 | 0.7918 | 0.7726 | 0.7982 |
| | Digital music | 0.0100 | 0.6345 | 0.7179 | 0.7783 | 0.7872 | 0.7860 | 0.7903 | 0.7958 |
| | Books | 0.0100 | 0.4753 | 0.5223 | 0.5785 | 0.5862 | 0.5787 | 0.5756 | 0.6016 |
| | Movies and TV | 0.0100 | 0.6284 | 0.6460 | 0.7230 | 0.7304 | 0.7357 | 0.7274 | 0.7493 |
| MRR@10 | CDs and Vinyl | 0.0200 | 0.4289 | 0.4735 | 0.5024 | 0.5147 | 0.5182 | 0.5128 | 0.5235 |
| | Digital Music | 0.0200 | 0.3295 | 0.3855 | 0.4452 | 0.4483 | 0.4574 | 0.4392 | 0.4587 |
| | Books | 0.0200 | 0.2943 | 0.3237 | 0.3268 | 0.3449 | 0.3457 | 0.3372 | 0.3489 |
| | Movies and TV | 0.0200 | 0.3376 | 0.3651 | 0.3869 | 0.3992 | 0.4023 | 0.3880 | 0.4087 |
| NDCG@10 | CDs and Vinyl | 0.0232 | 0.4966 | 0.5227 | 0.5693 | 0.5683 | 0.5702 | 0.5718 | 0.5727 |
| | Digital Music | 0.0215 | 0.3628 | 0.4436 | 0.5148 | 0.5259 | 0.5348 | 0.5327 | 0.5442 |
| | Books | 0.0153 | 0.3052 | 0.3396 | 0.3569 | 0.3496 | 0.3664 | 0.3572 | 0.3602 |
| | Movies and TV | 0.0186 | 0.3896 | 0.4060 | 0.4552 | 0.4682 | 0.4852 | 0.4615 | 0.4796 |

no movies or TV shows related to book content are output. This shows that LightGCN can only roughly capture the item attributes that users prefer, while our method can more accurately and comprehensively capture item information in the source domain, and through this method, recommend items for users that they may be interested in.




















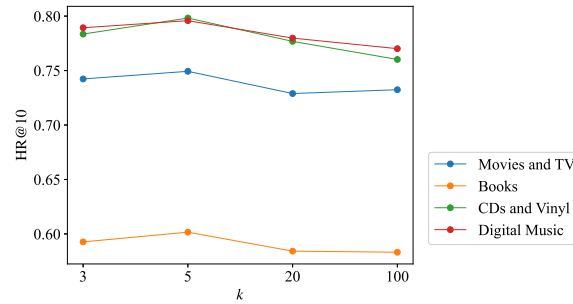
| Training Input (Books) | | | | | Recommended Output (Movies and TV) | | | | | |
|---|---|---|---|---|---|--|---|---|---|---|
|  |  |  |  |  | LightGCN |  |  |  |  |  |
| (a) | (b) | (c) | (d) | (e) | | (j) | (k) | (l) | (m) | (n) |
|  |  |  |  | OM |  |  |  |  |  | |
| (f) | (g) | (h) | (i) | | (o) | (p) | (q) | (r) | (s) | |

FIGURE 4. The items (Books) used as training input and the recommendation results (Movies and TV)

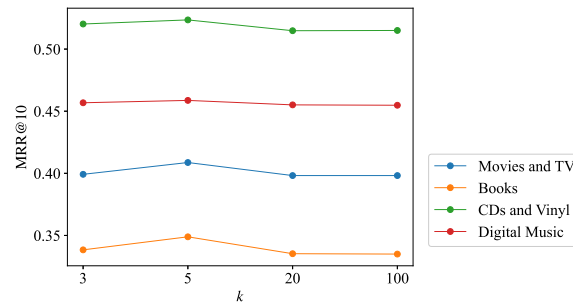
3.3. Influence of parameter k on the performance of the algorithm. The graph optimization module in our method optimizes the graph by detecting and deleting outliers, where the parameter k indicates that the detection and removal of outliers is carried out in the k -th neighborhood of the central node. In the four data domains, the relationship between the value of the parameter k and the number of detected outliers and the proportion of outliers are shown in Table 3. Analyzing the data in the table, we can conclude that no matter which data domain is in, as the assignment of k increases, the number of outliers and the proportion of outliers decreases. This is because when the value of k increases, the calculation range of the average density increases. During the calculation of the average density, as the number of participating nodes increases, the number of calculated outliers decreases accordingly.

TABLE 3. Number of outliers removed and their ratios

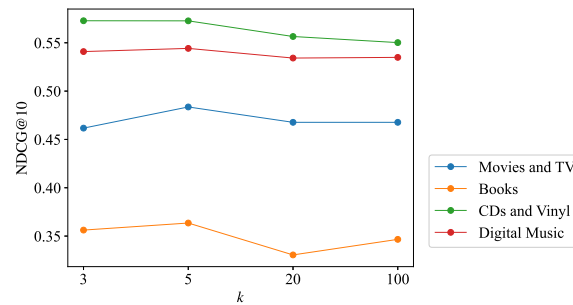
| k | Books | | Movies and TV | | CDs and Vinyl | | Digital Music | |
|-----|----------|-------|---------------|-------|---------------|-------|---------------|-------|
| | outliers | ratio | outliers | ratio | outliers | ratio | outliers | ratio |
| 3 | 13,061 | 4.87% | 2,261 | 4.59% | 2,658 | 4.76% | 157 | 4.42% |
| 5 | 9,560 | 3.58% | 1,704 | 3.46% | 2061 | 3.69% | 116 | 3.28% |
| 20 | 4497 | 1.68% | 807 | 1.64% | 904 | 1.62% | 54 | 1.54% |
| 100 | 3,096 | 1.17% | 551 | 1.12% | 692 | 1.24% | 36 | 1.02% |



(a) Hit Ratio



(b) Mean Reciprocal Rank



(c) Normalized Discounted Cumulative Gain

FIGURE 5. Algorithm performance under different values of parameter k

Figure 5 illustrates the performance comparison results of our method for various values of the parameter k . Analyzing the data in the figure, we can conclude that when the value of k is 3, the performance of the algorithm is poor. The reason is that the number of deleted nodes is too large, and too many nodes including features of users and items are lost, resulting in insufficient input data features during the training process. When the value of k is 20 and 100, the performance of the algorithm is also poor, because the number of deleted outliers is too small, too much redundant information is retained, and

some nodes with poor representation of embedded features are also retained, so it is not conducive to recommending training. When the assignment of k is 5, the performance of the algorithm is the best. At this time, the number of deleted nodes and the number of retained nodes are balanced to achieve the best algorithm effect.

4. Conclusion. In this study, we presented a novel algorithm for cross-domain recommendation by performing outlier removal on the graph in the latent space. The algorithm can efficiently remove negative nodes in the graph, and remove nodes that cannot adequately represent users' embedding features and items' embedding features in the graph, thereby boosting recommended accuracy of the algorithm. In addition, the bi-directional feature transfer module is applied, which enables the algorithm to better capture user preference attributes, thereby further improving the performance of the algorithm. On numerous datasets, efficacy of our method is demonstrated when compared to the most cutting-edge approaches currently available.

REFERENCES

- [1] Z.-H. Deng, L. Huang, C.-D. Wang, J.-H. Lai, and S. Y. Philip, "Deepcf: A unified framework of representation learning and matching function learning in recommender system," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, pp. 61–68.
- [2] H. Zhao and X. Wang, "Bi-group bayesian personalized ranking from implicit feedback," in *Proceedings of the 2nd International Conference on Computer Science and Software Engineering*, 2019, pp. 35–39.
- [3] Y. Wang, Y. Liu, H. Ma, Q. Ma, and Q. Ding, "The research of identity authentication based on multiple biometrics fusion in complex interactive environment," *Journal of Network Intelligence*, vol. 4, no. 4, pp. 124–139, 2019.
- [4] M. Tsagkias, T. H. King, S. Kallumadi, V. Murdock, and M. de Rijke, "Challenges and research opportunities in ecommerce search and recommendations," *ACM SIGIR Forum*, vol. 54, no. 1, pp. 1–23, 2021.
- [5] X. Xie, F. Sun, X. Yang, Z. Yang, J. Gao, W. Ou, and B. Cui, "Explore user neighborhood for real-time e-commerce recommendation," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 2464–2475.
- [6] C.-M. Chen, S. Lv, J. Ning, and J. M.-T. Wu, "A genetic algorithm for the waitable time-varying multi-depot green vehicle routing problem," *Symmetry*, vol. 15, no. 1, p. 124, 2023.
- [7] A. L. H. P. Shaik, M. K. Manoharan, A. K. Pani, R. R. Avala, and C.-M. Chen, "Gaussian mutation-spider monkey optimization (gm-smo) model for remote sensing scene classification," *Remote Sensing*, vol. 14, no. 24, p. 6279, 2022.
- [8] A. Rossi, D. Barbosa, D. Firmani, A. Matinata, and P. Merialdo, "Knowledge graph embedding for link prediction: A comparative analysis," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 15, no. 2, pp. 1–49, 2021.
- [9] 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.
- [10] T.-Y. Wu, J. C.-W. Lin, U. Yun, C.-H. Chen, G. Srivastava, and X. Lv, "An efficient algorithm for fuzzy frequent itemset mining," *Journal of Intelligent & Fuzzy Systems*, vol. 38, no. 5, pp. 5787–5797, 2020.
- [11] T.-Y. Wu, J. C.-W. Lin, Y. Zhang, and C.-H. Chen, "A grid-based swarm intelligence algorithm for privacy-preserving data mining," *Applied Sciences*, vol. 9, no. 4, p. 774, 2019.
- [12] H.-B. Ma, X.-G. Chen, Y.-L. Wang, and P.-J. Ji, "Efficient face attribute editing method based on gan," *Journal of Network Intelligence*, vol. 6, no. 3, pp. 646–655, 2021.
- [13] Y. Ma, Y. Peng, and T.-Y. Wu, "Transfer learning model for false positive reduction in lymph node detection via sparse coding and deep learning," *Journal of Intelligent & Fuzzy Systems*, vol. 43, no. 2, pp. 2121–2133, 2022.
- [14] P. Ji, H. Ma, Q. Ma, and X. Chen, "A novel method to generate pseudo-random sequence based on gan," *Journal of Network Intelligence*, vol. 7, no. 1, pp. 222–230, 2022.

- [15] P. Kadli and B. Vidyavathi, “Cross domain sentiment classification techniques: A review,” *International Journal of Computer Applications*, vol. 181, no. 37, pp. 13–20, 2019.
- [16] C. Zhao, C. Li, and C. Fu, “Cross-domain recommendation via preference propagation graphnet,” in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 2165–2168.
- [17] F. Zhu, C. Chen, Y. Wang, G. Liu, and X. Zheng, “Dtcd: A framework for dual-target cross-domain recommendation,” in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1533–1542.
- [18] C. Gao, X. Chen, F. Feng, K. Zhao, X. He, Y. Li, and D. Jin, “Cross-domain recommendation without sharing user-relevant data,” in *The World Wide Web Conference*, 2019, pp. 491–502.
- [19] F. Zhu, Y. Wang, C. Chen, G. Liu, and X. Zheng, “A graphical and attentional framework for dual-target cross-domain recommendation,” in *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, 2020, pp. 3001–3008.
- [20] Y. Wang, H. Wu, H. Ma, Q. Ma, and Q. Ding, “Speech denoising method based on improved least squares gan,” *Journal of Network Intelligence*, vol. 5, no. 3, pp. 113–121, 2020.
- [21] 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.
- [22] M. Liu, J. Li, G. Li, and P. Pan, “Cross domain recommendation via bi-directional transfer graph collaborative filtering networks,” in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 885–894.
- [23] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations*, 2015, pp. 1–15.
- [24] J. J. McAuley and J. Leskovec, “From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews,” in *Proceedings of the 22nd International Conference on World Wide Web*, 2013, pp. 897–908.
- [25] B. Loni, Y. Shi, M. Larson, and A. Hanjalic, “Cross-domain collaborative filtering with factorization machines,” in *European Conference on Information Retrieval*, 2014, pp. 656–661.
- [26] G. Hu, Y. Zhang, and Q. Yang, “Conet: Collaborative cross networks for cross-domain recommendation,” in *Proceedings of the 27th ACM International Conference on Information & Knowledge Management*, 2018, pp. 667–676.
- [27] C. Zhao, C. Li, R. Xiao, H. Deng, and A. Sun, “Catn: Cross-domain recommendation for cold-start users via aspect transfer network,” in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 229–238.