Cross Domain Recommendation via Bi-directional Transfer Graph Collaborative Filtering Networks

Meng Liu, Jianjun Li*
School of Computer Science and Technology, Huazhong University of Science and Technology, China
{sunshinel,jianjunli}@hust.edu.cn

Guohui Li
School of Software Engineering, Huazhong University of Science and Technology, China
guohuili@hust.edu.cn

Peng Pan
School of Computer Science and Technology, Huazhong University of Science and Technology, China
panpeng@hust.edu.cn

ABSTRACT

Data sparsity is a challenge problem that most modern recommender systems are confronted with. By leveraging the knowledge from relevant domains, the cross-domain recommendation technique can be an effective way of alleviating the data sparsity problem. In this paper, we propose a novel **Bi**-directional **T**ransfer learning method for cross-domain recommendation by using **G**raph **C**ollaborative **F**iltering network as the base model (BiTGCF). BiTGCF not only exploits the high-order connectivity in user-item graph of single domain through a novel feature propagation layer, but also realizes the two-way transfer of knowledge across two domains by using the common user as the bridge. Moreover, distinct from previous cross-domain collaborative filtering methods, BiTGCF fuses users' common features and domain-specific features during transfer. Experimental results on four couple benchmark datasets verify the effectiveness of BiTGCF over state-of-the-art models in terms of bi-directional cross domain recommendation.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Transfer learning*.

KEYWORDS

Recommender Systems; Collaborative Filtering; Transfer Learning; Graph Convolution Network

ACM Reference Format:
Meng Liu, Jianjun Li, Guohui Li, and Peng Pan. 2020. Cross Domain Recommendation via Bi-directional Transfer Graph Collaborative Filtering Networks. In *The 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3412012

1 INTRODUCTION

With the rapid increase of commodity types and quantities, personalized recommendation, which can predict the purchase intention of users, has become one of the most important services on the Internet. The personalized recommendation aims to predict a group of items that users are more likely to purchase in the future by fully employing uses' historical interactions.

Collaborative filtering (CF) is a widely used method [18, 27] for personalized recommendation, which learns the recommender model based on interaction history of similar users or items. Generally speaking, the key component in CF models is to learn the latent features (embeddings) of users and items effectively and then perform the prediction based on these embeddings. The traditional CF methods, represented by matrix factorization (MF), obtain the latent factors of users and items by factorizing the user-item interactive matrix [26]. Neural CF models, such as NeuMF [10], replace inner product by multiple neural network layers to obtain effectively matching function [5, 6, 23, 25]. Due to the powerful nonlinearity fitting ability of neural network, neural CF models in general can get better fitting results and have gradually become the mainstream.

In recent years, inspired by the success of graph convolutional networks (GCN) in effectively extracting features in non-Euclidean spaces, some researchers try to exploit the user-item bipartite graph structure by propagating embeddings on it, aiming at achieving more effective embeddings [1, 4, 14, 31]. For example, Wang *et al.* [29] proposed NGCF, which follows the same propagation rules as in GCN (including feature transformation, neighborhood aggregation and nonlinear activation) to capture the high-order connectivity between users and items by stacking multiple feature propagation layers, and achieves promising results. Recently, He *et al.* [9] found two common designs in GCNs, transformation function and nonlinear activation, have no positive effect on collaborative filtering or even degrade the performance. They proposed LightGCN, which greatly simplifies the design of NGCF but can yield better performance. In general, the integration of higher-order neighbor information makes GCN-based methods a great success. However, due to the large number of items in real life, recommender systems inevitably face the problem of data sparsity, which has become the main factor limiting the effectiveness of existing models.

An effective solution to the data sparsity problem is transferring knowledge [13] from other related domains by transfer learning. In real life, a user inevitably interacts with multiple domains to meet the demand of her life. When the interaction history is less in domain A, it is natural to consider getting some common knowledge from correlated domain B that includes more data. In recent years, cross-domain collaborative filtering (CDCF) [2, 11, 12, 15, 22, 28] has attracted increasing research attention. But like every coin

*Jianjun Li is the corresponding author.

CIKM '20, October 19–23, 2020, Virtual Event, Ireland
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3412012

has two sides, the correlations between domains make CDCF possible, the differences between domains also render it difficult to transfer knowledge. Early on, CodeBook Transfer (CBT) [15] was proposed to first compress the dense rating matrix of auxiliary domain into cluster-level rating pattern, called codebook, by orthogonal nonnegative matrix tri-factorization (ONMTF), and then realize knowledge transfer by sharing the codebook. Later, some variants of CBT that follow the similar transfer mechanism have been proposed [7, 21, 24]. This kind of method does not require the users of two domains to overlap. Unlike CBT's two-stage migration, Collective matrix factorization (CMF) [28] collectively factorizes the rating matrix of two domains with the same users (or items), and transfers knowledge by sharing the users' (or items') latent features [28]. This is an effective way to refine users' features in single domain with users' features learned from two domains. On this basis, some improvements have been proposed [17, 19]. For example, CoNet [11] takes neural network as the basic model and uses cross connections unit to improve the learning of matching functions in the current domain, while PPGN [35] extracts more effective common users' features by using GCN on the joint interaction graph of two domains.

Despite their effectiveness, most current CDCF methods only focus on refining user representations by sharing better common features, but without considering users' domain-specific features, which may limit the effect of transfer when the domain-specific features take a major proportion. Take Figure 1 for an example, there are two domains: Film and Book. Some of the user features in these two domains should be domain-specific, for instance, in the film domain, a user shows her preference for music, frame, etc., which is not available in the user's features in the book domain. But in existing CDCF methods, after transferring (sharing) user's features, the user's preferences in the two domains are exactly the same. Consequently, the matched movies based on the shared features may not be satisfactory.

In view of the limitation of existing CDCF methods and the powerful feature extracting ability of GCN, in this work, we propose a novel **Bi**-direction **T**ransfer learning method for cross-domain recommendation by using **G**raph **C**ollaborative **F**iltering network as the base model (BiTGCF). The major difference between BiTGCF with previous work lies in: (1) A new feature propagation module. Inspired by LightGCN, we remove the non-linear activation function and the transformation matrices in our GCN model for collaborative filtering, which greatly reduces the number of model parameters, as well as the risk of model overfitting. But different from LightGCN, we retain the inner product operation, the self-connection operation and the layer combination manner. We argue that these operations are beneficial for increasing the feature flow among nodes, which in turn can boost the recommendation performance. The experimental study validates our conjecture. (2) A bi-directional feature transfer module. Compared with previous CDCF methods, our model takes into account domain-specific features in different domains when refining user features. A simple yet effective balancing mechanism is designed to balance user's common features and domain-specific features. Through the bidirectional knowledge transfer between the two domains, our model can improve the recommendation performance of both domains simultaneously. The main contributions of this paper are as follows,

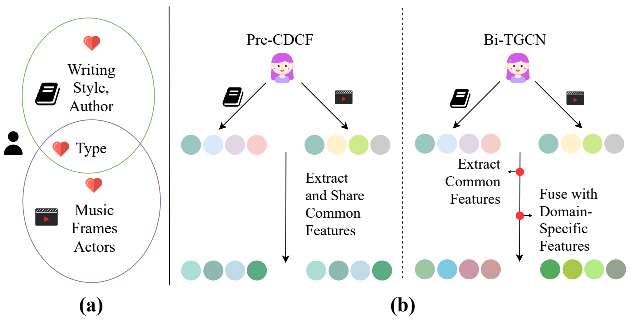


Figure 1: (a) Different preference of user in book and movie domain; (b) Different ways to extract user's feature in different domains: Previous CDCFs vs BiTGCF. The circle with the same color in (b) represents common feature.

- By using graph collaborative filtering network as the base model, we propose BiTGCF, a novel bi-directional transfer learning model for cross-domain recommendation. In BiTGCF, we design a new feature propagation module, which borrows the idea from LightGCN to simplify the feature propagation model in a more reasonable manner. Moreover, we propose a novel knowledge transfer module, which extends the flow of features from in-domain to inter-domain, and more importantly, considers the integration of uses' common features and domain-specific features.
- We empirically demonstrate that the proposed model BiTGCF outperforms the state-of-the-art approaches (for both single and cross-domain) on four couple cross-domain datasets. In addition, extensive experiments are conducted to verify the effectiveness and applicability of our feature transfer module.

The rest of this paper is organized as follows: Section 2 discusses related work; Section 3 formally defines the research problem and briefly reviews two representative recommendation models based on GCN; Section 4 details our proposed model; Section 5 presents the experimental study; Finally, Section 6 draws a conclusion.

2 RELATED WORK

2.1 Model-based CF Methods

Collaborative filtering (CF) is a commonly used technology in modern recommender systems that parameterizes users and items as embeddings and reconstructs the interactive history to learn the embedding parameters. The core of CF lies in how to design the model so that it can learn more effective embeddings. Earlier CF models, such as matrix factorization (MF), projects the user (or item) ID into embedding space, and models the matching relationship between user and item via inner product. The development of neural network provides a new idea for learning projection and matching function in CF models. For example, NeuMF [10] uses the stacked fully connection layer to replace the inner product in MF, and DMF [33] replaces the linear projection method in MF with the stacked full connection layer. Recently, researchers have found that different historical interactions contribute differently to the prediction of current interactions. To this end, attention mechanisms, such as ACF [3] and DeepICF [32], were introduced to automatically learn the importance of each historical interaction.

2.2 Graph Convolutional Networks based Recommendation

Inspired by the development of graph neural networks [14, 16, 30], there are some efforts on exploiting user-item interaction graph to infer user's preference. GC-MC [31] applies the graph convolutional network to exploit the connections between users and items when encoding interactive features. SpectralCF [36] utilizes a spectral convolution operation to explore all possible connectivity between users and items in the spectral domain. However, the eigen-decomposition in SpectralCF, which is a necessary step, is very time-consuming. Recently, Wang *et al.* [29] proposed the Neural Graph Collaborative Filtering (NGCF) framework to integrate GCN into the embedding process. By stacking multiple embedding propagation layers, NGCF can capture the collaborative signal in high-order connectivities between users and items. However, its designs are rather burdensome. LR-GCCF [4] removes the non-linear activation function to facilitate turning for large dataset. More importantly, it takes residual learning approach to explain the reason of concatenating all the layer's output. Later, LightGCN [9] simplifies NGCF by removing the operations such as activation function and transformation function that have no positive impact on collaborative filtering.

2.3 Transfer Learning and Cross Domain Collaborative Filtering

In recent years, transfer learning has emerged as a new learning framework to address the data sparsity problem by extracting and transferring knowledge from related domain. Cross Domain Collaborative Filtering (CDCF) is the application of transfer learning in recommendation, which focuses on how to transfer knowledge (features) in an effective way.

The way to transfer knowledge is various, such as Collective matrix factorization (CMF) [28] and codebook transfer [7, 15], which are based on Matrix Factorization (MF) applied in each domain. These approaches transfer interaction information from an auxiliary domain to improve the performance in a target domain with shallow model. Specifically, CMF jointly factorizes the rating matrix from two domains by sharing the user latent factors. This method effectively realizes the transfer and improvement of common user hidden features. The rise of deep learning contributes a lot to the development of CDCF, and some studies have tried to fuse CDCF with deep learning, such as CoNet [11] and its heterogeneous variants [19]. With MLP as the basic model, CoNet shares user features in the embedding process and completes the transfer of interaction features between the two domains through cross-mapping. DARec [34] extractes and transfer patterns from rating matrices in related domain, following the idea of domain adaptation. Later, DDTCDR [17] utilizes user information and items' metadata from online platform by using autoencoder, then adopts latent orthogonal mapping to extract user preferences over multiple domains. PPGN [35] adopts graph convolutional network to explore the high-order connectivity between users and items on the joint interaction graph of two domains, and then transfers knowledge by sharing user features. Compared with the shallow cross-domain matrix factorization models, the deep transfer methods generally exhibit better performance, due to their stronger feature extraction ability.

3 PRELIMINARY

3.1 Problem Definition

We consider two domains D_A and D_B. The set of users in both domains are shared, denoted by U (of size $m = |U|$). Let the set of items in D_A and D_B be I_A (of size $n_a = |D_\mathrm{A}|$) and I_B (of size $n_b = |D_\mathrm{B}|$), respectively. The purpose of bi-directional cross domain transfer is to improve the recommendation performance in both domains. We consider Top-N recommendation with implicit feedback in each domain. Let $R_\mathrm{A} \in \mathbb{R}^{m\times n_a}$ ($R_\mathrm{B} \in \mathbb{R}^{m\times n_b}$, resp.) denote the user-item interaction matrix of D_A (D_B, resp.) from users' implicit feedback, where an entry $r^\mathrm{A}_{ui} \in \{0, 1\}$ ($r^\mathrm{B}_{uj} \in \{0, 1\}$, resp.) is 1 if the interaction between user u on item i (item j, resp.) is observed, and 0 otherwise. The recommendation problem with implicit feedback is abstracted as learning a function to estimate the scores of unobserved entries in interaction matrix, which are later used for ranking. Specifically, for domain A,

$$\hat{r}^A_{ui} = f(u, i|\Theta) \tag{1}$$

where f is the interaction function, Θ represents all learnable parameters, and $\hat{r}^A_{ui}$ is the predicted score. For matrix factorization (MF) techniques, the matching function is the fixed dot product. For deep-learning based CF, such as NeuMF [10], the interaction is implemented by non-linear neural networks.

Obviously, extracting satisfactory embeddings for users and items is the key for better recommendation. Recently, GCN has shown its powerful ability in capturing the collaborative signal in high-order connectivities for more effective embedding learning. In view of this, in our transfer learning approach for cross-domain recommendation, each domain is also modeled by a graph convolutional network, and the GCN in both domains are jointly learned to improve the performance through bidirection high-order feature transfer. Before introducing our model in detail, we briefly review two representative recommendation models based on GCN, NGCF [29] and LightGCN [9], in the following subsection.

.

3.2 Brief Review of NGCF and LightGCN

The basic idea of GCN is to aggregate the features of neighbors so as to obtain better feature expression of nodes on the graph. GCN layer in general can be abstracted as:

$$\mathbf{e}_u^{(k+1)} = AGG(\mathbf{e}_u^{(k)}, \mathbf{e}_i^{(k)} : i \in \mathcal{N}_u). \tag{2}$$

where $AGG(\cdot)$ is an aggregation function such as weighted sum aggregator and mean aggregator, $\mathbf{e}_u^{(k)}$ and $\mathbf{e}_u^{(k)}$ respectively denote the refined embeddings of user u and item i after k layers propagation, and $\mathcal{N}_u$ denotes the first-hop neighbors of user u. In order to better understand the application of GCN in recommendation, we briefly introduce the embedding propagation in NGCF and LightGCN. Note that we only show the user feature propagation process in these two models, the item feature propagation process can be obtained analogously.

- Embedding Propagation Layer in NGCF:

$$\mathbf{e}_u^{(k+1)} = \sigma(\mathbf{W}_1^{(k)}\mathbf{e}_u^{(k)} + \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}(\mathbf{W}_1^{(k)}\mathbf{e}_i^{(k)} + \mathbf{W}_2^{(k)}(\mathbf{e}_i^{(k)} \odot \mathbf{e}_u^{(k)}))) \tag{3}$$

  where $\sigma(\cdot)$ is the non-linear activate function, $\mathbf{W}_1^{(k)}$ and $\mathbf{W}_2^{(k)}$ are trainable transformation matrices, $\mathcal{N}_u$ and $\mathcal{N}_i$ denote the first-hop neighbors of u and i, and $\odot$ is the element-wise product. Distinct from conventional graph convolutional networks that consider the contribution of $\mathbf{e}_i$ only, NDCG additionally encodes the interaction between $\mathbf{e}_i$ and $\mathbf{e}_u$ into the message being passed via $\mathbf{e}_i \odot \mathbf{e}_u$. The final user embedding is obtained by concatenating the output of L layers (i.e., $\{\mathbf{e}_u^{(1)}, \mathbf{e}_u^{(2)}, \ldots, \mathbf{e}_u^{(L)}\}$) with the initial embedding $\mathbf{e}_u^{(0)}$. Finally, the predictive score is obtained by conducting the inner product between the final user embedding and item embedding.
- Embedding Propagation Layer in LightGCN:

$$e_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}} e_i^{(k)} \tag{4}$$

  LightGCN greatly simplifies NGCF by removing the activation function and the transformation matrix (common in GCN but unfavorable for CF). Besides, it uses weighted sum in layer fusion to replace the self-connection in Equation (3) and concatenation in the fusion layer. Compared to NGCF, LightGCN reduces the risk of model overfitting to achieve better performance.

In this work, instead of utilizing the embedding propagation rule of NGCF or LightGCN directly, we will design our feature propagation layer. We borrow the idea of LigthGCN to simplify the design of GCN, in a more reasonable manner.

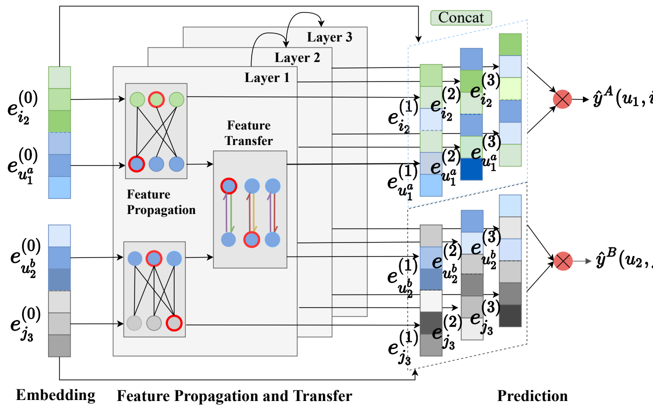
4 PROPOSED MODEL

We first introduce the overall structure of our BiTGCF model in Section 4.1, and then detail the two core components of BiTGCF, feature propagation and feature transfer, in Sections 4.2 and 4.3, respectively. Finally, we introduce the model training in Section 4.4.

4.1 Architecture Overview

As depicted in Figure 2, the proposed model BiTGCF mainly includes three modules: (1) an embedding layer that offers the initialization of user embeddings and item embeddings; (2) a feature propagation and transfer module (with multiple layers) that refines the initial embeddings of user and item by feature propagation in domain and feature transfer inter-domain; and (3) a prediction layer which concatenates the refined embeddings from different layers and outputs the probability that the given user-item pair is a positive interaction.

Embedding: This module maps the ID of a user u (an item i) into an embedding vector $\mathbf{e}_u^{(0)} \in \mathbb{R}^d$ ($\mathbf{e}_i^{(0)} \in \mathbb{R}^d$), where d denotes the embedding size. For ID embedding, this module can also be seen as building a parameter matrix as an embedding look-up table, which will be optimized in an end-to-end manner. Specifically, for domain



$e_{i_2}^{(0)}$
$e_{u_1^a}^{(0)}$
$e_{u_2^b}^{(0)}$
$e_{j_3}^{(0)}$
Layer 3
Layer 2
Layer 1
Feature Transfer
Feature Propagation
Concat
$e_{i_2}^{(1)}$ $e_{i_2}^{(2)}$ $e_{i_2}^{(3)}$
$e_{u_1^a}^{(1)}$ $e_{u_1^a}^{(2)}$ $e_{u_1^a}^{(3)}$
$e_{u_2^b}^{(1)}$ $e_{u_2^b}^{(2)}$ $e_{u_2^b}^{(3)}$
$e_{j_3}^{(1)}$ $e_{j_3}^{(2)}$ $e_{j_3}^{(3)}$
$\hat{y}^A(u_1, i_2)$
$\hat{y}^B(u_2, j_3)$
Embedding
Feature Propagation and Transfer
Prediction

Figure 2: An illustration of the architecture of BiTGCF (the arrowed lines present the flow of information), in which the red circle represents the current node according to the input.

A,

$$\mathbf{e}_{u^a}^{(0)} = \mathbf{P}^\top \mathbf{x}_{u^a}$$
$$\mathbf{e}_i^{(0)} = \mathbf{Q}^\top \mathbf{x}_i \tag{5}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are learnable parameter matrices of user and item, respectively, and $\mathbf{x}_u$ and $\mathbf{x}_i$ are one-hot encodings of the IDs of user $u \in U$ and item $i \in I_A$, respectively. Note that we use $\mathbf{e}_{u^a}^{(0)}$ and $\mathbf{e}_{u^b}^{(0)}$ to denote the embedding vectors of the same user u in D_A and D_B, respectively. Analogously, we can derive $\mathbf{e}_{u^b}^{(0)}$ and $\mathbf{e}_j^{(0)}$ for domain B, where $j \in I_B$.

Feature Propagation and Transfer: As shown in Figure 2, in this module, we feed $[\mathbf{e}_{u^a}^{(0)}, \mathbf{e}_i^{(0)}, \mathbf{e}_{u^b}^{(0)}, \mathbf{e}_j^{(0)}]$ through L graph convolution layers to refine the embeddings of users and items. This module consists of two components, feature (of both user and item) propagation within each domain, and feature (of only user) transfer between the two domains. We leverage the user-item interaction graphs to propagate and transfer embeddings as follows,

$$\begin{aligned}
\mathbf{e}_{u^a}^{(k+1)} &= f_\mathrm{T}^\mathrm{A}\left(f_\mathrm{P}^\mathrm{A}(\mathbf{e}_{u^a}^{(k)}), f_\mathrm{P}^\mathrm{B}(\mathbf{e}_{u^b}^{(k)})\right) \\
\mathbf{e}_{u^b}^{(k+1)} &= f_\mathrm{T}^\mathrm{B}\left(f_\mathrm{P}^\mathrm{A}(\mathbf{e}_{u^a}^{(k)}), f_\mathrm{P}^\mathrm{B}(\mathbf{e}_{u^b}^{(k)})\right) \\
\mathbf{e}_i^{(k+1)} &= f_\mathrm{P}^\mathrm{A}(\mathbf{e}_i^{(k)}) \\
\mathbf{e}_j^{(k+1)} &= f_\mathrm{P}^\mathrm{B}(\mathbf{e}_j^{(k)})
\end{aligned} \tag{6}$$

where $\mathbf{e}_{u^a}^{(k)}$ and $\mathbf{e}_i^{(k)}$ respectively denote the refined embeddings of u and i after k layers propagation in D_A, $\mathbf{e}_{u^b}^{(k)}$ and $\mathbf{e}_j^{(k)}$ respectively denote the refined embeddings of u and j after k layers propagation in D_B, $f_\mathrm{P}^\mathrm{A}(\cdot)$ and $f_\mathrm{P}^\mathrm{B}(\cdot)$ respectively denote the feature propagation function in D_A and D_B and will be defined in Section 4.2, $f_\mathrm{T}^\mathrm{A}(\cdot,\cdot)$ and $f_\mathrm{T}^\mathrm{B}(\cdot,\cdot)$ respectively denote the feature transfer function in D_A and D_B and will be defined in Section 4.3.

It is worth mentioning that in each layer, the feature transfer acts only on user features. Nevertheless, the refinement of item features can be achieved with the help of high-order connectivity between the two domains through the feature propagation module.

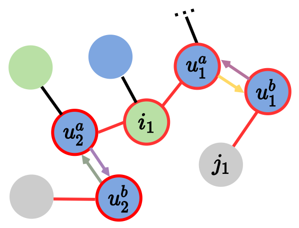


Figure 3: Illustration of Cross-domain Connectivity

Take the path marked in read in Figure 3 for example, i_1 gets part of message from its first-order (one-hop) neighbor u_1^a to refine its feature. Similarly, u_1^b also gets part of message from its first-order neighbor j_1 to refine its feature. Then by the transfer module, we can get the path $i_1 \leftarrow u_1^a \leftrightarrows u_1^b \leftarrow j_1$ (or $i_1 \rightarrow u_1^a \leftrightarrows u_1^b \rightarrow j_1$), indicating that the connectivity between j_1 and i_1 can be learned.

Prediction: After propagating and transferring with L layers, we can obtain multiple feature representations for users and items. The concatenation of multiple feature vectors learned from different order neighbors will result in a stronger and more robust joint representation for users and items. As such, we concatenate them to get the final representation vector for user u in D_A and item i as follows.

$$\begin{aligned} \mathbf{e}_{u^a} &= \mathbf{e}_{u^a}^{(0)} \| \cdots \| \mathbf{e}_{u^a}^{(L)} \\ \mathbf{e}_i &= \mathbf{e}_i^{(0)} \| \cdots \| \mathbf{e}_i^{(L)} \end{aligned} \tag{7}$$

where $\|$ denotes the concatenation operation. Finally, we adopt dot product, which is simple and non-parametric, to estimate the probability of the user interact with the target item,

$$\hat{r}_{ui}^{\mathrm{A}} = \hat{y}^A(u,i) = \sigma(\mathbf{e}_{u^a}^\top \mathbf{e}_i) \tag{8}$$

where $\sigma(\cdot)$ is the sigmoid function to map real values to probability. Note that domain B can be processed similarly to obtain $\hat{r}_{uj}^{\mathrm{B}}$.

4.2 Feature Propagation

Feature propagation aims to refine nodes' features on the current graph by aggregating message from neighbors. Inspired by NGCF [29] and LightGCN [9], we design a new feature propagation rule in this work. We remove the non-linear activation function and the transformation matrices in our GCN model for collaborative filtering. The effectiveness of such an operation has been verified in LightGCN [9]. But different from LightGCN, we retain the inner product operation, whose effectiveness has been verified by experiments in NGCF. Moreover, we use the same layer fusion operation as that in NGCF, which, according to [4], is equivalent to using residual prediction. Specifically, for a connected user-item pair (u, i) in domain A, we define the propagation function of u's and i's features as follows,

$$\begin{aligned} f_{\mathrm{P}}^{\mathrm{A}}(\mathbf{e}_{u^a}^{(k)}) &= \mathbf{e}_{u^a}^{(k)} + \sum_{i \in \mathcal{N}_{u^a}} \frac{1}{\sqrt{|\mathcal{N}_{u^a}||\mathcal{N}_i|}} \left(\mathbf{e}_i^{(k)} + \mathbf{e}_i^{(k)} \odot \mathbf{e}_{u^a}^{(k)}\right) \\ f_{\mathrm{P}}^{\mathrm{A}}(\mathbf{e}_i^{(k)}) &= \mathbf{e}_i^{(k)} + \sum_{u \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_{u^a}||\mathcal{N}_i|}} \left(\mathbf{e}_{u^a}^{(k)} + \mathbf{e}_{u^a}^{(k)} \odot \mathbf{e}_i^{(k)}\right) \end{aligned} \tag{9}$$

where $\mathcal{N}_{u^a}$ and $\mathcal{N}_i$ denote the first-hop neighbors of user u and item i, $\mathbf{e}_i^{(k)}$ and $\mathbf{e}_{u^a}^{(k)}$ are the representations of item i and user u passed from layer k, $\odot$ denotes the element-wise product, and the symmetric normalization term $\frac{1}{\sqrt{|\mathcal{N}_{u^a}||\mathcal{N}_i|}}$ follows the design of standard GCN to avoid the scale of embeddings increasing with propagation operations, which has also been used in NGCF and LightGCN. Note that for domain B, $f_{\mathrm{P}}^{\mathrm{B}}(\cdot)$ can be defined similarly.

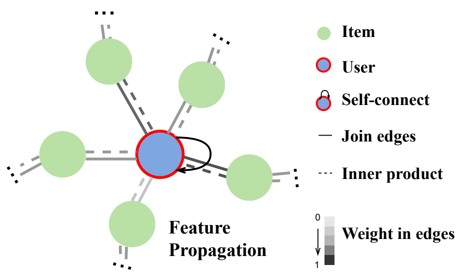


Figure 4: Illustration of Feature Propagation

The process of feature propagation is depicted in Figure 4. The message passed to the center node u along the connected edge from the neighbor nodes (items) includes two parts: (1) the normalized weighted feature; and (2) its inner product with the center node u. The former is the general operation of GCN to integrate information from the neighbor nodes, while the latter ensures that the larger the match scores between the user and the item, the greater the value passed to the center node. Finally, we consider self-connection of the center node u to retain the information of the original features.

4.3 Feature Transfer

Feature transfer is the key component of BiTGCF, which realizes bidirectional user feature transfer between D_A and D_B. As mentioned in Section 1, we take both common features and domain-specific features into consideration, which is different from existing CDCF methods that only consider learning common features. We define the feature transfer function of u in D_A and D_B as follows,

$$f_{\mathrm{T}}^{\mathrm{A}}(\cdot,\cdot) = \frac{1}{2}(C^{(k)} + A^{(k)}) \tag{10}$$

$$f_{\mathrm{T}}^{\mathrm{B}}(\cdot,\cdot) = \frac{1}{2}(C^{(k)} + B^{(k)}) \tag{11}$$

Specifically,

$$C^{(k)} = l_{u^a} f_{\mathrm{P}}^{\mathrm{A}}(\mathbf{e}_{u^a}^{(k)}) + l_{u^b} f_{\mathrm{P}}^{\mathrm{B}}(\mathbf{e}_{u^b}^{(k)}) \tag{12}$$

$$A^{(k)} = \lambda^a f_{\mathrm{P}}^{\mathrm{A}}(\mathbf{e}_{u^a}^{(k)}) + (1-\lambda^a) f_{\mathrm{P}}^{\mathrm{B}}(\mathbf{e}_{u^b}^{(k)}) \tag{13}$$

$$B^{(k)} = (1-\lambda^b) f_{\mathrm{P}}^{\mathrm{A}}(\mathbf{e}_{u^a}^{(k)}) + \lambda^b f_{\mathrm{P}}^{\mathrm{B}}(\mathbf{e}_{u^b}^{(k)}) \tag{14}$$

where l_{u^a} and l_{u^b} are user-related weight factors for domain A and B respectively, λ^a and λ^b are hyper-parameters within the range of $[0, 1]$ that are used to control the retention ratio of user features in the corresponding domain. Note that l_{u^a} (l_{u^b}) is calculated based on the ratio of the number of u's interacted items in D_A (D_B) to the number of items u interacted with in both domains, i.e.,

$$l_{u^a} = \frac{|\mathcal{N}_{u^a}|}{|\mathcal{N}_{u^a}| + |\mathcal{N}_{u^b}|} \tag{15}$$

$$l_{u^b} = 1 - l_{u^a} \tag{16}$$

It is clear to see that Equation (12) adopts the idea of feature propagation on the graph, which forms improved feature by aggregating the features from $f_{\mathrm{P}}^{\mathrm{A}}(\mathbf{e}_{u^a}^{(k)})$ and $f_{\mathrm{P}}^{\mathrm{B}}(\mathbf{e}_{u^b}^{(k)})$. In this way, we can regard $C^{(k)}$ as the common features derived for u under both D_{A} and D_{B}. Moreover, the design of the weight factors l_{u^a} and l_{u^b} indicates that more interaction data there is for u in D_{A} (or D_{B}), more features in D_{A} (or D_{B}) will contribute to the common features, and vice versa.

$A^{(k)}$ and $B^{(k)}$ are used to preserve the users' domain-specific features in D_{A} and D_{B} respectively. We use the hyper-parameters λ^a and λ^b to control the retention ratio. For example, when $\lambda^a = \lambda^b = 1.0$, it indicates that 100% of the user features in D_{A} and D_{B} is retained, so user features have the largest specificity in both domains. When $\lambda^a = \lambda^b = 0.5$, the specificity in users' features disappears and the same users have the same features in both domains, and the transfer module in BiTGCF becomes the same as that in existing CDCF methods.

Finally, we emphasis equally on the obtained common features and domain-specific features to balance them, as shown in Equations (10) and (11). This operation, though simple, is very effective in keeping the stability of the model performance, as validated in our experimental study.

4.4 Model Training

The deep model calculates the gradient to update the parameters through the loss function. Therefore, a suitable loss function should not only avoid the model falling into local optimization but also accelerate the model convergence. For recommender systems, two types of loss functions are widely used, point-wise which focuses on predicting scores more accurately, and pair-wise [8] which focuses on learning rank more accurately. In this paper, we consider point-wise loss function and leave the pair-wise version to our future work. The most commonly used point-wise loss is the squared loss (SE), but it is not suitable for implicit feedback. Hence, following several previous work, we employ the binary cross-entropy function as the loss function, which can be defined as,

$$L(\hat{r}_{ui}, r_{ui}) = -\sum_{(u,i)\in R^+\cup R^-} r_{ui}\log\hat{r}_{ui} + (1-r_{ui}\log(1-\hat{r}_{ui})) + \lambda\|\Theta\|_2^2 \quad (17)$$

where R^+ is the set of observed interaction history, and R^- is the set of randomly sample from unobserved interaction. λ controls the $L2$ regularization strength to prevent overfitting. $\Theta = \{E_u^0, E_i^0\}$ and E_u^0 (E_i^0) is the initial embedding matrix for all users (items). In order to improve the accuracy on both domains, we define the joint loss function as follows,

$$L_{join} = \min_{f_{\mathrm{T}}^{\mathrm{A}}, f_{\mathrm{P}}^{\mathrm{A}}, f_{\mathrm{T}}^{\mathrm{B}}, f_{\mathrm{P}}^{\mathrm{B}}} L(\hat{r}_{ui}^{\mathrm{A}}, r_{ui}^{\mathrm{A}}) + L(\hat{r}_{uj}^{\mathrm{B}}, r_{uj}^{\mathrm{B}}) \quad (18)$$

We adopt mini-batch Adam to optimize the model and update the parameters. Moreover, similar to NGCF, we introduce dropout mechanisms to prevent neural networks from overfitting. Specifically, we drop out the messages being propagated in Equation (9) with a probability in training, and disable this operation during testing. Note that BiTGCF is a bi-directional transfer mode in which data from two domains participate in the training at the same time, but are evaluated separately.

Table 1: Statistics of the datasets

Dataset	#Users	#Items	#Interactions	Density
Elec	3,325	39,463	118,879	0.091%
Cell	3,325	18,462	53,732	0.088%
Sport	9,928	32,310	102,540	0.032%
Cloth	9,928	41,303	97,757	0.024%
Sport	4,998	22,101	55,556	0.050%
Cell	4,998	14,618	47,444	0.065%
Elec	15,761	53,309	226,626	0.027%
Cloth	15,761	51,865	136,844	0.017%

5 EXPERIMENT

We first describe the experimental setup in Section 5.1, and then compare the proposed model with state-of-the-art methods in Section 5.2. To justify the effectiveness of the transfer learning module, we study its adaptability in Sections 5.3, as well as its performance under different data sparsity levels in Section 5.4. Finally, the impact factor analysis is presented in Section 5.5.

5.1 Experimental setup

5.1.1 Dataset. We evaluate our proposed model on real-world datasets from Amazon dataset[1], including two couple datasets, Electronics (**Elec** for short) & Cell Phones (**Cell** for short), and Accessories, Sports and Outdoors (**Sport** for short) & Clothing Shoes and Jewelry (**Cloth** for short). Moreover, in order to show that BiTGCF still has good transfer capability in domains with lower similarity, the cross pairing of the two groups, **Sport** & **Cell** and **Elec** & **Cloth**, are used as the third and fourth couple datasets. For the data in these four couple datasets, we first transform them into implicit data, where each entry is marked as 0 or 1, indicating whether the user has rated the item. Then, we filter the datasets to retain users with number of ratings greater than 5 and items with number of ratings greater than 10, and extract the overlapping users in both domains. Table 1 summarizes the detailed statistics of the four couple datasets.

5.1.2 Evaluation Protocol. We adopt the widely used leave-one-out evaluation method. Specifically, we take a random sample from each user's interaction history as the test set, and the remaining are utilized for training. Then we randomly select 99 items from each user's non-interacted items to form negative samples. The recommendation model would predict 100 records (99 negative samples and 1 positive sample) of the user and output top-N items. We use the commonly used Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) to evaluate the ranking performance. For both measures, we truncate the ranked list at 10. Hence, the HR measures whether the test item is present on the top-10 list, and the NDCG measures the ranking quality by assigning higher scores to hits at top ranks.

5.1.3 Compared Methods. We compare BiTGCF with both single domain and cross domain recommendation models. We leave out the comparison with DDTCDR [17] and DARec [34], because they

[1]http://jmcauley.ucsd.edu/data/amazon/

Table 2: Performance comparison in terms of HR and NDCG. Best performance is in boldface and the best baselines are underlined.

Dataset	Metrics	Single domain Methods				Cross domain Methods				Ours	
		BPRMF+	MLP+	NGCF+	LightGCN+	CMF	CDFM	CoNet	PPGN	GCF+	BiTGCF
Elec	HR	0.3471	0.4448	0.4665	0.4701	0.3894	0.3998	0.4484	0.4337	0.5871	**0.6036**
	NDCG	0.2224	0.2814	0.2969	0.2990	0.2457	0.2513	0.2890	0.2664	0.3682	**0.3767**
Cell	HR	0.4271	0.4848	0.4983	0.5185	0.4239	0.4117	0.4899	0.4386	0.6198	**0.6571**
	NDCG	0.2919	0.3108	0.3358	0.3253	0.2631	0.2599	0.3102	0.2773	0.4119	**0.4210**
Sport	HR	0.2784	0.3660	0.4251	0.4623	0.3108	0.3196	0.3743	0.3484	**0.5421**	0.5346
	NDCG	0.1778	0.2169	0.2681	0.2910	0.1837	0.1874	0.2254	0.2038	**0.3414**	0.3375
Cloth	HR	0.2250	0.3464	0.3723	0.4011	0.2930	0.2501	0.3442	0.3239	0.5101	**0.5242**
	NDCG	0.1348	0.2073	0.2359	0.2570	0.1612	0.1509	0.2090	0.1819	0.3067	**0.3113**
Sport	HR	0.2990	0.3663	0.3858	0.3978	0.3157	0.3345	0.3700	0.3316	0.5018	**0.5442**
	NDCG	0.2025	0.2368	0.2527	0.2498	0.1930	0.2163	0.2392	0.2260	0.3122	**0.3374**
Cell	HR	0.3315	0.4764	0.4926	0.4970	0.3918	0.4574	0.4824	0.4418	0.5486	**0.5654**
	NDCG	0.2211	0.3080	0.3173	0.3074	0.2286	0.1605	0.3134	0.2851	0.3566	**0.3709**
Elec	HR	0.2780	0.4877	0.5007	0.5086	0.4580	0.4106	0.4931	0.4633	0.5357	**0.5359**
	NDCG	0.1761	0.3191	0.3235	0.3288	0.3032	0.2817	0.3220	0.2920	0.3549	**0.3575**
Cloth	HR	0.2262	0.3562	0.3632	0.3662	0.3196	0.3017	0.3461	0.3221	0.4003	**0.4384**
	NDCG	0.1371	0.2160	0.2238	0.2199	0.1907	0.1801	0.2015	0.1934	0.2461	**0.2555**

both use Auto-encoder as a feature extractor for pre-training, which is different from the end-to-end form of the methods to be evaluated.

- **BPRMF** [26] is a classical single-domain model, which learns the user and item factors via matrix factorization and pairwise rank loss.
- **MLP** [10] is single-domain deep model, which uses deep neural networks to learn the matching function.
- **NGCF** [29] is a single-domain GCN based model. It first captures the high-order connectivity information in the embedding function by stacking multiple embedding propagation layers, and then concatenates the obtained embeddings and uses inner product to make prediction.
- **LightGCN** [9] is also a single-domain GCN based model evolved from NGCF. It simplifies the design in the feature propagation component by removing the non-linear activation and the transformation matrices. Moreover, it adopts a different layer combination strategy from NGCF.
- **CMF** [28] is a multi-relation learning approach which factorizes matrices of domains A and B simultaneously by sharing the user latent factors. It is a shallow model, which first jointly learns on two domains, and then optimizes the target domain.
- **CDFM** [20] is a cross-domain shallow model. It takes user interaction history from auxiliary domains as context to generate a recommendation on the target domain with factorization machines.
- **CoNet** [11] is a cross-domain deep model, which transfers knowledge across domains by cross connections between the base networks. It jointly learns on two domains and optimizes both domains simultaneously. Note that our model BiTGCF is different from CoNet in that it transfers user-item interaction features, while BiTGCF transfers user features.
- **PPGN** [35] is a cross-domain deep model, which fuses the interaction information of the two domains into a graph, and shares the features of users learned from the joint interaction graph by stacking multiple graph convolution layers. Finally, it inputs the learned embeddings to the domain-specific MLP structure to learn the matching function.
- **GCF** is a degenerate version of BiTGCF without the feature transfer layer. It is a single-domain GCN based model.

Note that in order to exclude the influence of loss function on the results, we change the loss function of NGCF and LightGCN from BPR to cross entropy, so as to unify the loss function among all the methods based on deep learning. Moreover, in order to more accurately evaluate the effect of our transfer module, for single-domain methods including BPRMF, MLP, NGCF, LightGCN and GCF, we use 'model-name+' to denote the same model but with mixed datasets as the training set. For example, given MLP+ and a couple dataset Elec&Cell, MLP+ will use Elec combining with Cell as the training set, and then test on Elec and Cell respectively.

5.1.4 Parameter Settings. For BPRMF, we use the BPRMF class in LightFM[2], a popular CF library, for training and vary the number of epoch from 0 to 40 to get the best result. For CMF, we use a Python version reference to the original Matlab code and the parameters are randomly initialized from Gaussian $\mathcal{N}(0, 0.01)$. For CDFM, we process and code the data according to the paper, then feed it to PyFM[3] for training. For MLP[4] and CoNet[5] with deep structure, we maintain the optimal configuration in their papers, the configuration of the hidden layers is $[64, 32, 16, 8]$ and the ratio of negative sampling is set to 4. For NGCF[6] and LightGCN[7], we use the published source code and only change the loss function from BPR to cross entropy. Moreover, we set the embedding propagation layer

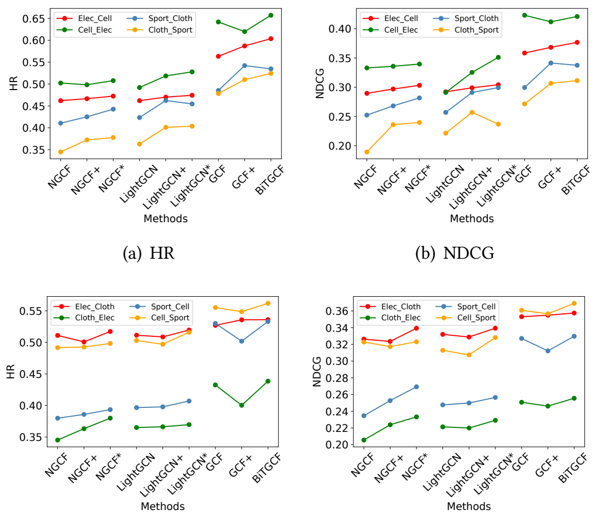
[2]https://github.com/lyst/lightfm
[3]https://github.com/coreylynch/pyFM
[4]https://github.com/hexiangnan/neural_collaborative_filtering
[5]http://home.cse.ust.hk/ghuac/
[6]https://github.com/xiangwang1223/neural_graph_collaborative_filtering
[7]https://github.com/kuandeng/LightGCN

as [64, 64, 64], the learning rate as 0.001, the size of mini batch as 1024, the ratio of negative sampling as 4, and the message dropout ratio as 0.1 for NGCF and 0 for LightGCN, respectively. For PPGN, we use the source code provided by the author and change the data pipeline. We also set the ratio of negative sampling as 4, and turn the number of GCN layers from 3 to 5 as the paper states. For GCF and BiTGCF, we implement them by TensorFlow, and use the same parameter settings as that of NGCF. We use the Xavier initializer to initialize the parameters of NGCF, LightGCN and our methods. Moreover, early stopping is performed if HR@10 on the test data does not increase for 5 successive epochs. Note that CoNet, PPGN and BiTGCF are bi-directional transfer models, which means we can obtain the results on a couple dataset at the same time, while the performance of single domain methods are obtained by first training on the mixed datasets and then evaluating separately on each dataset.

5.2 Performance Comparison

Table 2 shows the summarized results of our experiments on the four-couple datasets in terms of two metrics, HR@10 and NDCG@10. Due to space concern, the performance of the single-domain methods under the single dataset is not shown in Table 2. In fact, from our experiments, the performance of the single-domain models under mixed dataset in general is better than that under the single dataset, and partial results regarding this point will be reported in Section 5.3. From Table 2, we have the following key observations:

- For single domain methods, MLP+ consistently outperforms BPRMF+, demonstrating the importance of exploring nonlinear interaction relation between users and items. GCN-based methods, including NGCF+, LightGCN+ and GCF+, consistently outperforms MLP+. This validates the effect of mining high-order connectivities for better recommendation performance. The improvement of LightGCN+ over NGCF+ may come from the enhancement of its generalization ability, which is less prone to overfitting. Further, compared with NGCF+ and LightGCN+, GCF+ retains the inner product and the self-connection operation to appropriately increase the feature flow among nodes. This might be the reason that GCF+ can gain superior performance over LightGCN+.
- Cross domain methods *vs.* single domain methods. The superior performance of CMF and CDFM over BPRMF+ provides evidence that compared to directly mixing the two datasets, the transferred knowledge from auxiliary domain does improve the performance of the target domain. The performance of CoNet is better than MLP+ in most cases, which validates the effect of the cross connect unit, however the improvement is very limited than direct mixing the two datasets. Moreover, we can see that NGCF beats CoNet on almost all the datasets. This demonstrates that the transfer learning mechanism, if not well designed, is not as effective as mining high-order connectivities for better embeddings. Surprisingly, PPGN is worse than NGCF and even worse than CoNet. This might be caused by the datasets, since PPGN has been shown to outperform CoNet on CD&Music and Book&Movie in [35]. We guess the poor performance of PPGN on datasets with large distribution gaps might come from that it uses the same propagation layer on the joint graph of the two domains. BiTGCF achieves the best performance among the evaluated single-domain and cross-domain methods on all the datasets, which indicates that it is on the right track to exploit the high-order connections in-domain and inter-domain by using graph structures and fuse common features with domain-specific features through the feature transfer module.



(a) HR (b) NDCG

(c) HR (d) NDCG

Figure 5: The applicability of the transfer module.

5.3 The Applicability of the Transfer Module

In order to verify the applicability of our transfer module, we incorporate GCN-based models NGCF and LightGCN with our transfer module, and name the derived methods as NGCF* and LightGCN*. Figure 5 shows the results. In Figure 5(a) and 5(b), the two domains of the two couple datasets (Elec&Cell, Sport&Cloth) have higher similarity, while in Figure 5(c) and 5(d), the two domains of the two couple datasets (Elec&Cloth, Sport&Cell) have lower similarity. In order to indicate the pairing information of the current dataset, the expression of the 'target'_'source' domain is adopted. For example, Elec_Cell means the Elec dataset with the participation of Cell.

Comparing the first point with the second point in each method, we can observe a growth trend in Figure 5(a) and 5(b) for all the methods in most cases, as well as a downward trend in Figure 5(c) and 5(d). The reason might be that, the mixing of data in two similar domains increases the amount of training set, which promotes the training of the model and in turn improves the performance. On the contrary, when the two domains are not similar, simply mixing them may lead to model confusion, which makes it difficult to train and thus degrade the performance.

Comparing the third point with the first two in each method, we can conclude that the methods with transfer mechanism in general can achieve the best result. This validates the effectiveness of our transfer module. In summary, compared with directly mixing datasets, knowledge transfer yields to better performance.

Table 3: Statistics of splited datasets.

Dataset	Groups	Num≤	#Users	#Interactions	Density
Sport	G1	8	2769	17008	0.028%
	G2	14	1306	14207	0.049%
	G3	30	726	14422	0.090%
	G4	323	197	9919	0.23%
Cell	G1	6	2209	11930	0.037%
	G2	10	1638	13256	0.055%
	G3	20	893	12410	0.095%
	G4	155	258	9848	0.26%

Table 4: Results on Sport_Cell. The improvement is the result of BiTGCF to the best one from GCF and GCF+.

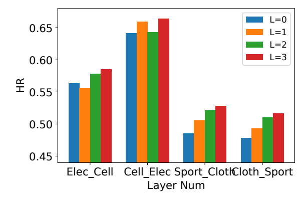
Metrics	Methods	G1	G2	G3	G4
HR	GCF	0.5081	0.5253	0.5840	0.6447
	GCF+	0.4778	0.5061	0.5606	0.5935
	BiTGCF	0.5314	0.5330	0.5923	0.6374
	Improv.	4.59%	1.47%	1.42%	-1.13%
NDCG	GCF	0.3061	0.3307	0.3747	0.4267
	GCF+	0.3001	0.3125	0.3536	0.4035
	BiTGCF	0.3230	0.3326	0.3760	0.4301
	Improv.	4.87%	1.03%	1.54%	1.75%

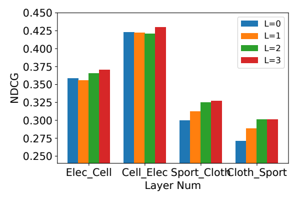
Table 5: Results on Cell_Sport. The improvement is the result of BiTGCF to the best one from GCF and GCF+.

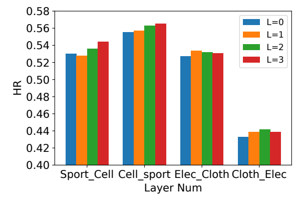
Metrics	Methods	G1	G2	G3	G4
HR	GCF	0.5147	0.5549	0.6036	0.7093
	GCF+	0.5093	0.5516	0.6011	0.6852
	BiTGCF	0.5283	0.5720	0.6103	0.686
	Improv.	2.64%	3.08%	1.11%	-3.28%
NDCG	GCF	0.3340	0.3530	0.4032	0.4910
	GCF+	0.3290	0.3506	0.4015	0.4789
	BiTGCF	0.3419	0.3636	0.4251	0.4787
	Improv.	2.37%	3%	5.43%	-2.51%

5.4 Performance of the Transfer Module *w.r.t* Data Sparsity Levels

The number of user interactions is an important factor that affects the recommendation performance. In order to test the influence of data sparsity levels, we divided the dataset into groups with different sparsity according to the number of user interactions. Specifically, with the principle of minimizing the difference in the number of interactions between each group, all users are divided into four groups G1, G2, G3 and G4 in the order of increasing number of interactions. Table 3 shows the splited results of Sport & Cell, in which users with the number of interactions no more than 8, 14, 30 and 323 are divided into G1, G2, G3 and G4, respectively. From Table 4, we can find that the improvement achieved in the first two groups (e.g., 4.59% and 1.47%) are more significant than that of the last two (e.g.,1.42% and -1.13%). The result indicates that the feature transfer module can help improve recommendation performance for relatively inactive user (with less interaction items). A similar trend can be observed from Table 5.


(a) HR


(b) NDCG


(c) HR

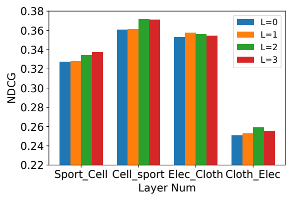
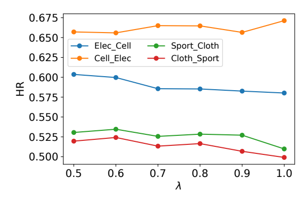
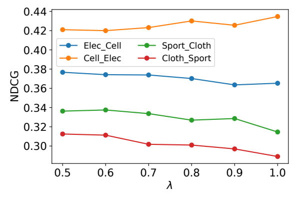
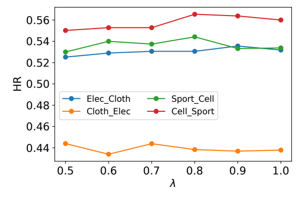

(d) NDCG

Figure 6: Effect of Transfer Layer Number


(a) HR


(b) NDCG


(c) HR

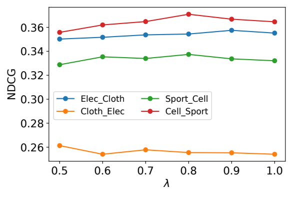

(d) NDCG

Figure 7: Effect of λ

5.5 Impact Factor Analysis

5.5.1 Number of feature transfer layers. The feature transfer module is the core of BiTGCF, and its effect is verified by changing the number of transfer layers with fixed number of features propagation layer. In particular, we vary the layer number L in the range of $\{0, 1, 2, 3\}$. More specifically, based on the single-domain model GCF (with three propagation layers but non transfer layers), the transfer layer is added from the top layer. For example, when $L = 1$, one feature transfer component is added in the last feature propagation layer; when $L = 2$, two feature transfer components are added in the last two feature propagation layers.

From Figure 6, we can see that with the increase of L, the performance of BiTGCF also increases. The reason may come from that the feature transfer module communicates different domains, enabling the model to mine higher-order relationships between nodes by stacking multiple features propagation and transfer layers. Moreover, it can be seen from Figure 6(c) and 6(d) that when $L = 3$, the performance of BiTGCF degrades on some datasets. This indicates that mining for higher-order connectivities is not always beneficial for the performance improvement. However, when $L = 0$, the performance of BiTGCF is always the worst, which demonstrates the effect of our transfer module.

5.5.2 Hyper-parameters λ^a & λ^b. In this set of experiments, we analyze the influence of hyper-parameters λ^a and λ^b on the performance. For simplicity, we always consider a single hyper-parameter $\lambda = \lambda^a = \lambda^b$. Figure 7 shows the results, with a step size of 0.1, ranging in $[0.5, 1]$. Figure 7(a) and 7(b) show the results of BiTGCF on Elec & Cell and Sport & Cloth, respectively. The overall best result is achieved when λ is around 0.5 and 0.6. The results on Sport & Cell and Elec & Cloth are shown in Figure 7(c), 7(d), where the best result appears when $\lambda = 0.8$. The result implicitly reflects that, when the similarity between the two domains is low, it is necessary to retain more domain-specific features.

6 CONCLUSION

In this paper, we proposed BiTGCF for Top-N cross-domain recommendation by combining the idea of high order feature propagation in graph structure with transfer learning. Inspired by NGCF and LightGCN, we designed a new feature propagation module, which simplifies the feature propagation in a more reasonable manner to reduce the risk of model over-fitting but still preserve some non-linearity to boost the recommendation performance. Moreover, we took the first time to consider the domain-specific features in different domains when refining user features, and designed a simple but effective balancing mechanism to balance user's common features and domain-specific features. By combining inter-domain feature transfer with in-domain feature propagation, our model realizes more efficient representation and transfer of users' common features and their integration with domain-specific features. Remarkable performance improvements on several benchmark datasets demonstrate the effectiveness of our BiTGCF model.

ACKNOWLEDGMENTS

The work was supported by the National Natural Science Foundation of China under Grant No. 61672252, and the Fundamental Research Funds for the Central Universities under Grant No. 2019kfyXKJC021.

REFERENCES

[1] Stephen Bonner, Ibad Kureshi, John Brennan, Georgios Theodoropoulos, Andrew Stephen McGough, and Boguslaw Obara. 2019. Exploring the Semantic Content of Unsupervised Graph Embeddings: An Empirical Study. *Data Science and Engineering* 4, 3 (2019), 269–289.

[2] Bin Cao, Nathan Nan Liu, and Qiang Yang. 2010. Transfer Learning for Collective Link Prediction in Multiple Heterogenous Domains. In *Proceedings of ICML*. Omnipress, 159–166.

[3] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention. In *Proc. of SIGIR*. ACM, 335–344.

[4] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. In *Proc. of AAAI*. 27–34.

[5] Zhi-Hong Deng, Ling Huang, Chang-Dong Wang, Jian-Huang Lai, and Philip S. Yu. 2019. DeepCF: A Unified Framework of Representation Learning and Matching Function Learning in Recommender System. In *Proceedings of AAAI*. 61–68.

[6] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative Memory Network for Recommendation Systems. In *Proc. of SIGIR*. 515–524.

[7] Sheng Gao, Hao Luo, Da Chen, Shantao Li, Patrick Gallinari, and Jun Guo. 2013. Cross-domain recommendation via cluster-level latent factor model. In *Proceedings of ECML-PKDD*. 161–176.

[8] Ruining He and Julian McAuley. 2016. VBPR: visual bayesian personalized ranking from implicit feedback. In *Proceedings of Thirtieth AAAI*. 144–150.

[9] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proc. of SIGIR*. 639–648.

[10] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of WWW*. 173–182.

[11] Guangneng Hu, Yu Zhang, and Qiang Yang. 2018. Conet: Collaborative cross networks for cross-domain recommendation. In *Proceedings of CIKM*. 667–676.

[12] Guangneng Hu, Yu Zhang, and Qiang Yang. 2019. Transfer Meets Hybrid: A Synthetic Approach for Cross-Domain Collaborative Filtering with Text. In *Proceedings of WWW*. ACM, 2822–2829.

[13] Liang Hu, Jian Cao, Guandong Xu, Longbing Cao, Zhiping Gu, and Can Zhu. 2013. Personalized recommendation via cross-domain triadic factorization. In *Proc. of WWW*. 595–606.

[14] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of ICLR (Poster)*.

[15] Bin Li, Qiang Yang, and Xiangyang Xue. 2009. Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In *Proceedings of Twenty-First International Joint Conference on Artificial Intelligence*. 2052–2057.

[16] Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. 2019. DeepGCNs: Can GCNs Go As Deep As CNNs?. In *Proceedings of ICCV*. 9266–9275.

[17] Pan Li and Alexander Tuzhilin. 2020. DDTCDR: Deep Dual Transfer Cross Domain Recommendation. In *Proc. of WSDM*. ACM, Houston, USA, 331–339.

[18] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 1 (2003), 76–80.

[19] Jian Liu, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Fuzhen Zhuang, Jiajie Xu, Xiaofang Zhou, and Hui Xiong. 2019. Deep Cross Networks with Aesthetic Preference for Cross-domain Recommendation. *CoRR* abs/1905.13030 (2019). http://arxiv.org/abs/1905.13030

[20] Babak Loni, Yue Shi, Martha A. Larson, and Alan Hanjalic. 2014. Cross-Domain Collaborative Filtering with Factorization Machines. In *Proceedings of 36th European Conference on Advances in Information Retrieval Research*. 656–661.

[21] Zhongqi Lu, Weike Pan, Evan Wei Xiang, Qiang Yang, Lili Zhao, and Erheng Zhong. 2013. Selective Transfer Learning for Cross Domain Recommendation. In *Proceedings ICDM*. SIAM, 641–649.

[22] Zhongqi Lu, Yin Zhu, Sinno Jialin Pan, Evan Wei Xiang, Yujing Wang, and Qiang Yang. 2014. Source Free Transfer Learning for Text Classification. In *Proceedings of AAAI*. AAAI Press, 122–128.

[23] Jingwei Ma, Jiahui Wen, Mingyang Zhong, Weitong Chen, and Xue Li. 2019. MMM: Multi-source Multi-net Micro-video Recommendation with Clustered Hidden Item Representation Learning. *Data Science and Engineering* 4, 3 (2019), 240–253.

[24] Orly Moreno, Bracha Shapira, Lior Rokach, and Guy Shani. 2012. TALMUD: transfer learning for multiple domains. In *Proceedings of CIKM*. ACM, 425–434.

[25] ThaiBinh Nguyen and Atsuhiro Takasu. 2018. NPE: Neural Personalized Embedding for Collaborative Filtering. In *Proceedings of IJCAI*. 1583–1589.

[26] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proc. of UAI*. 452–461.

[27] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proc. of WWW*. 285–295.

[28] Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of SIGKDD*. ACM, 650–658.

[29] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proc. of the 42nd ACM SIGIR*. 165–174.

[30] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In *Proceedings of ICML*. 6861–6871.

[31] Yuexin Wu, Hanxiao Liu, and Yiming Yang. 2018. Graph Convolutional Matrix Completion for Bipartite Edge Prediction. In *Proceedings of IC3K*. 49–58.

[32] Feng Xue, Xiangnan He, Xiang Wang, Jiandong Xu, Kai Liu, and Richang Hong. 2019. Deep Item-based Collaborative Filtering for Top-N Recommendation. *ACM Trans. Inf. Syst.* 37, 3 (2019), 33:1–33:25.

[33] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep Matrix Factorization Models for Recommender Systems. In *Proceedings of IJCAI*. 3203–3209.

[34] Feng Yuan, Lina Yao, and Boualem Benatallah. 2019. DARec: Deep Domain Adaptation for Cross-Domain Recommendation via Transferring Rating Patterns. In *Proceedings of IJCAI*. 4227–4233.

[35] Cheng Zhao, Chenliang Li, and Cong Fu. 2019. Cross-Domain Recommendation via Preference Propagation GraphNet. In *Proceedings of CIKM*. 2165–2168.

[36] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S. Yu. 2018. Spectral collaborative filtering. In *Proc. of ACM RecSys*. 311–319.