Hierarchical Representation Learning for Bipartite Graphs

Chong Li[1*], **Kunyang Jia**[3], **Dan Shen**[2,3], **C.J. Richard Shi**[1] and **Hongxia Yang**[3]
[1]University of Washington
[2]University of South Florida
[3]DAMO Academy, Alibaba Group
{chongli, cjshi}@uw.edu, {kunyang.jky, d.shen, yang.yhx}@alibaba-inc.com

Abstract

Recommender systems on E-Commerce platforms track users' online behaviors and recommend relevant items according to each user's interests and needs. Bipartite graphs that capture both user/item feature and use-item interactions have been demonstrated to be highly effective for this purpose. Recently, graph neural network (GNN) has been successfully applied in representation of bipartite graphs in industrial recommender systems. Providing individualized recommendation on a dynamic platform with billions of users is extremely challenging. A key observation is that the users of an online E-Commerce platform can be naturally clustered into a set of communities. We propose to cluster the users into a set of communities and make recommendations based on the information of the users in the community collectively. More specifically, embeddings are assigned to the communities and the user information is decomposed into two parts, each of which captures the community-level generalizations and individualized preferences respectively. The community structure can be considered as an enhancement to the GNN methods that are inherently flat and do not learn hierarchical representations of graphs. The performance of the proposed algorithm is demonstrated on a public dataset and a world-leading E-Commerce company dataset.

1 Introduction

The online retail market is developing at a rapid pace and customers are actively looking for more engaging and highly personalized retail experiences. To achieve success in a highly dynamic market, E-Commerce businesses must be able to stay one step ahead of their customers by predicting what customers are looking for, based on their past click-through behaviors, shopping history, and product preferences. However, in a complex environment of a world-leading E-Commerce platform, how to predict user-specific and unique behaviors among billions of online customers is quite challenging. A widely-adopted methodology is to represent the users and items as nodes in a *bipartite graph* [Grbovic and Cheng, 2018; Ying *et al.*, 2018a]. The relationships between the nodes are treated as edges in the graph. Then a GNN based method [Hamilton *et al.*, 2017] encodes the nodes of the resulted graph as low-dimensional *embeddings*, which capture the node attributes as well as the relationships between the nodes. With this formulation, the recommendation task could be transformed as the link prediction problem.

[*]Work done as an intern at DAMO Academy, Alibaba Group

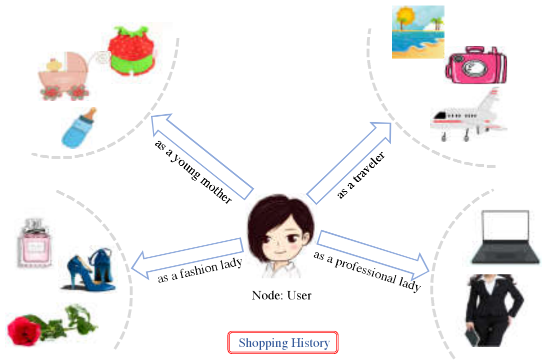

Figure 1: A user's shopping pattern. The young lady belongs to four communities based on her various social roles. The shopping pattern reflects her community belonging as well as her personal preferences.

While much effort has been made to address the idiosyncrasies of individual users in an E-Commerce recommender system, we make the note that users can be naturally clustered into a set of *communities* based on their demographic, income level, occupation, among other factors. Figure 1 illustrates a user's community-level shopping preferences. This lady plays four community roles: a business professional, a young mother, a traveler, and a fashion lady. As a mother, she routinely purchases infant care products. This shopping behavior is highly related to her community role as a young mother. Making recommendations for a certain community is arguably a less challenging task [Grbovic, 2017], as we have access to significantly larger amount of data concerning all the users in this community collectively. With the number of communities being dramatically lower than the number of users, we could also afford to carry out more elaborated analyses and perform more sophisticated feature engineering for the communities. This observation is among the motivations of a series of works on hierarchical graph neural networks [Chen *et al.*, 2018; Zitnik and Leskovec, 2017; Ying *et al.*, 2018b], in which the network learns to cluster the users into communities. However, striking a delicate balance between community-level generalization and user-specific preference can be a major challenge. Going back to the example in Figure 1, the lady may prefer high-end products with dark colors. These individualized preferences are not reflected in the communities that she belongs to. Most of the existing works in hierarchical graph networks ignore the user-specific information that is not captured by the user's communities. For this reason, while hierarchical graph network has been demonstrated to be effective in graph classification tasks, it has not been widely applied in online personalized recommender systems.

To tackle such challenges, we present *Bi-HGNN* (**B**ipartite **H**ierarchical **G**raph **N**eural **N**etwork), a recommender system for the E-Commerce platform that effectively utilizes the users' community-level generalization *as well as* their individualized preferences. *Bi-HGNN* enhances the current GNN methods that are inherently flat and do not learn hierarchical representations of graphs. To be more specific, each community is assigned an embedding, which is jointly trained with other parameters of the GNN. The distances between a particular user and the communities are computed by a neural network based distance function. This distance metric allows us to softly assign each user into a few communities. We decompose user information into two orthogonal spaces, each of which represents information captured by community-level generalization and individualized user preference respectively. This decomposition is our primary technical contribution. *Bi-HGNN* is capable of automatically clustering users into communities in an end-to-end fashion. To demonstrate its scalability, we have also deployed *Bi-HGNN* on a world-leading E-Commerce platform. Our numerical experiments demonstrate that *Bi-HGNN* consistently outperforms state-of-the-art recommendation algorithms.

The remainder of this paper is organized as follows: in Section 2, we review related works. Section 3 provides a detailed description of the proposed *Bi-HGNN* framework. Experimental results on both the public dataset and a real-world E-Commerce dataset are shown in Section 4. Section 5 concludes the paper.

2 Related Work

As conventional graph analytic methods often suffer from scalability issue, a new paradigm, called *graph embedding* (GE) [Perozzi *et al.*, 2014; Tang *et al.*, 2015; Grover and Leskovec, 2016; Abu-El-Haija *et al.*, 2018; Bojchevski and Günnemann, 2018], paves an efficient yet effective way to address large scale graph representation problems. Specifically, GE converts the nodes and/or edges of the graph into low-dimensional embeddings such that vital information of the graph are preserved. Based on this idea, a series of GNNs have been proposed, including Structure2Vec [Ribeiro *et al.*, 2017], GCN [Kipf and Welling, 2017], and GraphSAGE [Hamilton *et al.*, 2017]. Notably, GraphSAGE proposes an inductive framework to encode node features as well as neighborhood information into the node embedding. However, the aforementioned GNN methods are inherently flat and do not learn hierarchical representations of graphs.

Hierarchical graph representation has also been previously proposed. [Chen *et al.*, 2019] uses the non-backtracking operator to yield improvements on hard community detection problems. Since the user generally belongs to several communities on an E-Commerce platform, the hard assignment is not suitable. DiffPool [Ying *et al.*, 2018b] assigns a user into communities using soft assignment. However, a fundamental limitation of DiffPool is that it requires explicitly expressing and computing with the adjacent matrix of the graph. This is computationally prohibitive for recommender system that operates on graphs with tens of millions of nodes. More importantly, DiffPool focuses on the homogeneity of the clusters while ignoring the node diversity. In this paper, we also propose an extension of DiffPool that preserves its core features yet can scale to real-world recommender systems presented in the experiment section. [Zhu *et al.*, 2018] presents a tree-based model to construct a hierarchical structure of items. The computation complexity to chose an item to be recommended is logarithmic w.r.t. the number of items.

3 *Bi-HGNN* Model

Let $G(U, I, E)$ be a bipartite graph, where U is a set of user nodes and I is a set of item nodes and E is a set of edges that records user-to-item and item-to-item relationships. u and i represents individual user and item nodes respectively. x is the raw feature of the nodes. $\mathcal{N}$ is the neighborhood of a node. $\mathcal{N}_u$ is user u's neighbors and $\mathcal{N}_i$ is item i's neighbors. e_u and e_i represents user and item embedding.

A GNN method [Hamilton *et al.*, 2017; Ying *et al.*, 2018a] encodes the raw feature and the neighborhood information of the nodes as embeddings. The user embedding and item embedding are trained in an end-to-end fashion where the supervision signal is whether a user interacted with a particular item. When a new user joins the platform, his/her user embedding is computed and recommendations for the new user is solely based on the user embedding and the item embeddings.

As shown in Figure 2, we first encode the raw features of the users and items with the Wide and Deep [Cheng *et al.*, 2016]. The raw features could be either numerical or categorical.

$$e_u^{(W)} = \mathrm{WD}_u(x_u), \tag{1}$$
$$e_i^{(W)} = \mathrm{WD}_i(x_i), \tag{2}$$

where $\mathrm{WD}_u(.)$ and $\mathrm{WD}_i(.)$ are Wide and Deep module for user and item respectively. We then feed $e_u^{(W)}$ and $e_i^{(W)}$ as h_u^0 and h_i^0 into GraphSAGE [Hamilton *et al.*, 2017] to encode neighborhood information as follows:

$$h_{\mathcal{N}_u}^t = \mathrm{aggregate}_u^t(\{h_i^{t-1}, i \in \mathcal{N}_u\}), \tag{3}$$
$$h_u^t = \sigma\left(W_u^t \cdot \mathrm{concat}(h_u^{t-1}, h_{\mathcal{N}_u}^t)\right), \tag{4}$$

where $\sigma(.)$ is the sigmoid function, W_u^t is a trainable parameter, t is the number of hops of the neighbors. h_u^t is normalized

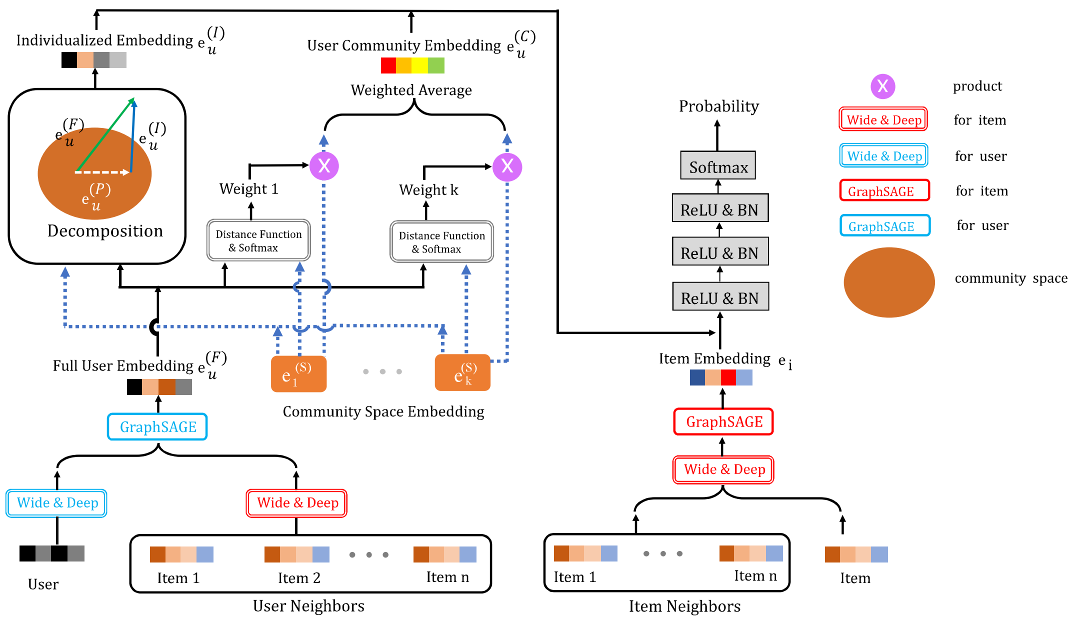

Figure 2: Framework of *Bi-HGNN*. Left side outputs *user-community* and *individualized* embeddings. Since user and item raw features are generally sparse, wide and deep is used here for feature encoding. Based on the user's historical behaviors, n items are selected as his/her neighbors to generate *full* user embedding with GraphSAGE. *Full* embedding is decomposed into *user-community* and *individualized* embeddings, which is illustrated in the upper left of the figure within the dashed square. *User-community* embedding is a weighted linear combination of the community space embedding, where coefficients are determined by the distance between the user and *community space* embedding. With both the user and item embeddings, the classifier computes the probability of the user interacting with the item.

to yield the *full user embedding*:

$$e_u^{(F)} = \frac{h_u^t}{\|h_u^t\|}. \tag{5}$$

Item embedding can be computed in a similar way.

$$h_{\mathcal{N}_i}^t = \text{aggregate}_i^t(\{h_j^{t-1}, j \in \mathcal{N}_i\}), \tag{6}$$
$$h_i^t = \sigma\left(W_i^t \cdot \text{concat}(h_i^{t-1}, h_{\mathcal{N}_i}^t)\right), \tag{7}$$

$$e_i = \frac{h_i^t}{\|h_i^t\|}. \tag{8}$$

The *full user embedding* and item embedding computed as above have been demonstrated to be highly effective in industrial recommender systems [Ying *et al.*, 2018a; Grbovic and Cheng, 2018].

3.1 Community Embedding for Clustering

The primary technical contribution of this work is in clustering the users into communities and perform recommendation based on the user's community generalization as well as individualized preference. The clustering is achieved by (1) assigning low-dimensional embeddings to a set of communities, and (2) devising a distance function that quantifies the "closeness" between a user and a community using their embeddings.

As shown in the upper left part of Figure 2, *community space embedding*, denoted by $e_1^{(S)}, \cdots, e_k^{(S)}$, is a trainable parameter that is randomly initialized to be an orthonormal matrix. Here we require the number of the communities is smaller than the dimension of the full user embedding. Given a *community space embedding* $e^{(S)}$ and a *full user embedding* $e_u^{(F)}$, the user's distance to the community is computed as

$$w_{u,l} = f_d(e_u^{(F)}, e_l^{(S)}) \tag{9}$$

where $l \in \{1, \cdots, k\}$ and $f_d(\cdot)$ is the distance function that takes the two embeddings as argument and yield a scalar in interval $[0, 1]$. We shall elaborate on the design this distance function later in this section.

A fundamental assumption of this work is that the user's community-level generalization can be expressed as a weighted linear combination of community space embeddings, with the coefficients being the distance between the user and the community. Intuitively, if a user's online behavior closely matches the collective behavior of a community, the embedding of this particular community should have a large weight in the user's community-level generalization. Formally, given the distance between user and community, the weighted average of *community embedding* yields the *user-community embedding*

$$e_u^{(C)} = \sum_{l=1}^{k} w_{u,l}\, e_l^{(S)}. \tag{10}$$

Ideally we would like to a user to belong to only a few communities. In other words, only a few of the coefficients in could take non-zero value. This is achieved by passing a user's distance to the communities through a SoftMax function. The *user-community embedding* in Equation (3.1) captures community behavior but not the user's individualized preferences.

In general, a user's information can not be fully captured by a linear combination of the community space embeddings. This is because the number of the communities is smaller than the dimension of the full user embedding, thus it is impossible to perfectly reconstruct the full user embedding using a linear combination of community space embeddings. Intuitively, the community-level information of the user can be represented by a linear combination of community space embeddings, while the user-specific individualization has to be captured in the orthogonal space of the community space embeddings.

Formally, let S be the space spanned by community space embedding $e^{(S)}$, $S = \text{span}\{e_1^{(S)}, \cdots, e_k^{(S)}\}$. Projecting the full user embedding $e_u^{(F)}$ onto space S yields *user-community embedding* $e_u^{(P)}$, which is the best possible approximation of $e_u^{(F)}$ using linear combination of $e^{(S)}$.

$$e_u^{(P)} = \text{proj}\left(e_u^{(F)}, S\right), \tag{11}$$

In the orthogonal space, the *individualized embedding* $e_u^{(I)}$ that captures the individualized preference of the user is simply

$$e_u^{(I)} = e_u^{(F)} - e_u^{(P)}. \tag{12}$$

We assume that $e_u^{(I)}$, $e_u^{(F)}$ and $e_u^{(P)}$ are of the same dimension.

The projection in Equation 11 can be computed in various ways. We choose compute this projection using singular value decomposition (SVD) [Golub and Reinsch, 1971], for the reason that SVD is a differentiable operation that can be readily incorporated in a deep neural network. Decompose matrix $C = [e_1^{(S)}, \cdots, e_k^{(S)}]$ with SVD as $C = U\Sigma V^T$, where U and V are the singular vectors and Σ is a diagonal matrix with singular values on the diagonal. The column vectors of matrix V, denoted by $\{\zeta_l \mid l \in \{1, \cdots, k\}\}$, are orthonormal bases of the space spanned by community space embedding $e^{(S)}$. With the bases of the space, the projection in Equation 11 can be easily computed as

$$e_u^{(P)} = < e_u^{(F)}, \zeta_1 > \zeta_1 + \cdots + < e_u^{(F)}, \zeta_k > \zeta_k, \tag{13}$$

where $< .,. >$ is the inner product between two vectors.

Note that the inner product $< .,. >$ is the coefficients of the weighted linear combination. In other words, the inner product $< .,. >$ is essentially the distance function that we have discussed earlier in this section. In this work, we used a neural network as the distance function. Alternatively, it is possible to use the dot product, which is efficient to compute but less expressive.

We summarize the notations of embeddings discussed in this section as follows:

The resulted user-community embedding $e_u^{(P)}$, individualized embedding $e_u^{(I)}$, and item embedding are fed into a classifier. The classifier computes the probability of the user interaction with a particular item, given their embeddings.

$e_u^{(F)}$	Full user embedding due to GraphSage
$e_i^{(F)}$	Item embedding due to GraphSage
$e_l^{(S)}$	Community Space Embedding (Embedding assigned to a community)
$e_u^{(P)}$	User-community embedding (Project of $e_i^{(F)}$ on the space spanned by $e_l^{(S)}$)
$e_u^{(I)}$	Individualized embedding (Individual preference cannot be captured by $e_u^{(P)}$)

3.2 Algorithm Framework and Loss Function

In the training phase, let s_u^+ and s_u^- be positive and negative samples for user u respectively. Given user u, $y_{u,i}$ is the ground truth label. If user u has the interaction with item i, then $y_{u,i} = 1$, otherwise $y_{u,i} = 0$. For user u, $\hat{y}_{u,i}$ is the predicted label for item i. The likelihood function is derived as

$$\prod_u \left\{ \prod_{i \in s_u^+} P(\hat{y}_{u,i} = 1) \prod_{i \in s_u^-} P(\hat{y}_{u,i} = 0) \right\} \tag{14}$$

which yields the loss function

$$-\sum_u \sum_{i \in s_u^+ \cup s_u^-} \{y_{u,i} \log P(\hat{y}_{u,i} = 1) + (1 - y_{u,i}) \log P(\hat{y}_{u,i} = 0)\}. \tag{15}$$

Bi-HGNN is summarized in Algorithm 1.

Algorithm 1 *Bi-HGNN* Algorithm

Input: bipartite graph $G(U, I, E)$, user raw feature x_u, item raw feature x_i.
Output: Probability that user u would be interested in item i

```
1: Initialize $e_l^{(S)}, l = 1, \cdots, k$;
2: Wide and deep for user in Equation (1): $e_u^{(W)} \leftarrow x_u, \forall u \in U$;
3: Wide and deep for item in Equation (2): $e_i^{(W)} \leftarrow x_i, \forall i \in I$;
4: $e_u^{(F)} \xleftarrow{(3),(4),(5)} \text{GraphSAGE}\left(e_u^{(W)}, \{e_i^{(W)}, i \in \mathcal{N}_u\}\right)$;
5: $e_u^{(C)} \xleftarrow{(9),(3.1)} \text{weightedAvg}\left(e_u^{(F)}, \{e_1^{(S)}, \cdots, e_k^{(S)}\}\right)$;
6: $e_u^{(I)} \xleftarrow{(11),(12)} \text{decompose}\left(e_u^{(F)}, \{e_1^{(S)}, \cdots, e_k^{(S)}\}\right)$;
7: $e_i \xleftarrow{(6),(7),(8)} \text{GraphSAGE}\left(e_i^{(W)}, \{e_j^{(W)}, j \in \mathcal{N}_i\}\right)$;
8: Feed $e_u^{(I)}$, $e_u^{(C)}$ and $e_i$ into the classifier to generate the probability of $\hat{y}_{u,i} = 1$ in Equation (14) as
     $P(\hat{y}_{u,i} = 1) \leftarrow \text{classifier}\left(e_u^{(I)}, e_u^{(C)}, e_i\right)$
9: return Probability that the user would be interested in an item.
```

4 Numerical Experiments

In this section, we describe the setup of our numerical experiments. We report our results on the widely used MovieLens dataset[1] as well as the real-world online shopping data from a world-leading E-Commerce company. The baseline methods we compared against are GraphSAGE [Hamilton *et al.*, 2017] and DiffPool [Ying *et al.*, 2018b]. GraphSAGE has been demonstrated to be an effective yet scalable GNN algorithm in a number of applications. We shall demonstrate that *Bi-HGNN* outperforms GraphSAGE by considering the community information. DiffPool, which is closely related to our method, recursively clusters nodes in a graph into communities. As previously discussed in Section 2, DiffPool suffers from severe scalability issues. For comparison purpose, we developed a modified version of DiffPool, named DiffPool-S (S for scalable), that is scalable to graphs with tens of millions of nodes, yet preserves the key characteristics of DiffPool. To be more specific, DiffPool-S matches DiffPool in the following two aspects: (1)A user is assigned to the communities based on a pre-selected distance function that matches specific task requirement; (2)Once a user is clustered into a community, majority of their individualized information is lost while only keeping abstracted community level information.

Additionally, we perform ablation studies on *Bi-HGNN* to evaluate the importance of its individual building blocks, with details summarized as follows:

- Bi-GNN: In this study, we remove the orthogonal decomposition step in *Bi-HGNN* as in Equation (12). Formally, the inputs to the classifier are $e_u^{(F)}$, $e_u^{(C)}$, and e_i.
- O-GNN: This variant is formulated similarly as DiffPool-S, except that we explicitly require the *user-community embedding* to be orthonormal in every training step. To satisfy the orthonormal requirement, let W_{CO} be a trainable parameter with proper size, and $W_{CO} = U\Sigma V^T$ be the SVD of W_{CO}. Since SVD is a differentiable operation, the constructed orthonormal V is trainable.

We choose not to consider methods based on matrix factorization [Zhang *et al.*, 2006] or Jaccard similarity [Moulton and Jiang, 2018], since they are quite shallow and usually fail to capture high-order non-linearity or interactions in real-world recommender systems. Also, we do not consider transductive methods that lack the ability to make inferences for unobserved samples, thus impractical in practice.

4.1 Movielens Dataset

The MovieLens dataset contains around 6,000 user's numerical rating of approximately 3,000 films. The raw features of the users include gender, age, and occupation, while the raw features of the movies are year and genre. We split the dataset into training and testing by the time stamp. For every user, we use the first 70% of the data to predict the rest 30% of future behaviors: whether this particular user would be interested in watching the movie. We compared *Bi-HGNN* to the baseline methods on four key classification metrics on the MovieLens dataset. The results are summarized in Table 1.

[1]https://grouplens.org/datasets/movielens

Method	Accuracy	F1	AUC (ROC)	AUC (PR)
GraphSAGE	85.7%	24.1%	63.1%	17.5%
Bi-GNN	84.9%	23.8%	63.1%	17.4%
DiffPool	79.3%	24.0%	62.5%	16.7%
O-GNN	76.0%	23.2%	61.6%	16.1%
Bi-HGNN	**89.5%**	**42.9%**	**82.0%**	**40.1%**

Table 1: Performance comparison of *Bi-HGNN* to the baseline methods on the MovieLens dataset.

In a link prediction setting, the Movielens dataset contains the positive samples, i.e. the movies that a user has actually watched. To train a classifier, we also need to generate negative samples. Since in the real-world online shopping scenario where a user is only interested in a tiny fraction of the items, we created the training and testing set in a way that the positive samples and negative samples are highly imbalanced (negative to positive ratio is 10). For such imbalanced classification problems, we shall pay attention not only to the classification accuracy, but also to the robustness of the classifier. We evaluated with accuracy measures including F1, area under curve (AUC) of Receiver Operating Characteristics (ROC) and precision. As shown in Table 1, the performance of Bi-GNN is almost identical to that of the baseline GraphSAGE. One may find this result a bit counter-intuitive, since compared to GraphSAGE, Bi-GNN could utilize *user-community embedding* $e_u^{(C)}$ and corresponding distances as auxiliary sources of information. However, we argue that $e_u^{(C)}$ is inferred from $e_u^{(F)}$ and simple concatenation of the two does not bring extra information. This observation highlights the importance of our proposed orthogonal decomposition, which effectively forces the neural network to encode useful community information in the training process.

The performance of DiffPool-S and O-GNN is worse than other methods. This is expected since in these two methods, all the individualized information that is not captured by the community embedding are completely lost. This illustrates the challenge of applying existing hierarchical GNNs in recommender systems, where highly specific information of individual users is required. By effectively capturing both community generalization and user-specific information, *Bi-HGNN* achieves superior performances both in terms of accuracy and robustness.

4.2 E-Commerce Dataset

We also performed experiments on real online shopping data from a world-leading E-Commerce platform. To preserve anonymity, we use Company to represent the E-Commerce company that provided this dataset. We randomly sampled approximately 4.4 million active users who interacted with approximately 5.9 million items from the Company's platform on 02/14/2019 as the training dataset. The transaction data from 02/15/2019 are randomly sampled to construct the testing dataset. The users and the items that appeared in the testing dataset are not necessarily in the training dataset. The statistics of the underlying bipartite graph is summa-

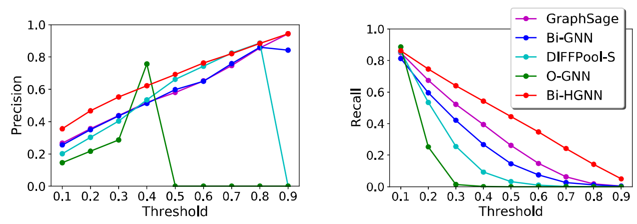

Figure 3: Precision and Recall comparison. The metrics are computed at thresholds evenly spaced between 0.1 and 0.9.

	Users	Items	User-Item Edges	Item-Item Edges
Train	4.4 M	5.9 M	76.4 M	102.9 M
Test	5 M	6.2 M	92.6 M	110 M

Table 2: Statistics of the bipartite graph for the E-Commerce dataset (in millions).

rized in Table ??. A user is described by 21 categorical features (e.g. purchasing power, geolocation) and 5 continuous features (e.g. number of transactions, user-item preference scores), while an item is characterized by 8 categorical features (e.g. item category, shipping costs) and 22 continuous features (e.g. click-through rate, number of clicks). A user-to-item edge is established if the user clicked the item in the past 15 days. The weight of a user-to-item edge is computed by a time-decaying preference score, which depends on the action of the user (click, add to cart, or purchase). We used the collaborative filtering algorithm [Ekstrand *et al.*, 2011] to calculate the weight of an item-to-item edge. In the training phase, we randomly sample 10 neighbors of a node based on the weight of the connecting edges as the neighbourhood in GraphSAGE.

The comparison of our proposed method to the baseline methods are summarized in Table 3. We also plotted the curve of precision and recall computed at thresholds evenly spread out between 0.1 and 0.9 in Figure 3. Again, O-GNN performs the worst. Also, as shown in Figure 3, the classifier broke down at threshold 0.5 in the precision metric. This phenomenon suggests that in hierarchical GNN, forcing the community embedding to be orthonormal across all the training steps is likely to be too stringent. With such a requirement, the stochastic gradient descent algorithm probably missed the opportunity to escape from local minimums.

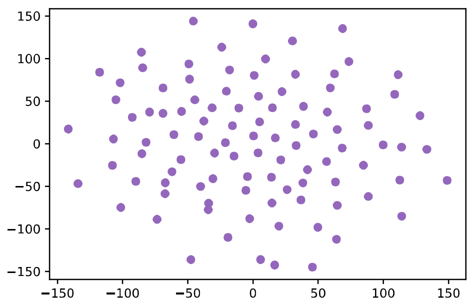

Figure 4: t-SNE plot of the trained *community space embedding*. The dimension of the *community space embedding* is 256 and the number of community is 100.

Method	Accuracy	F1	AUC (ROC)	AUC (PR)
GraphSAGE	88.3%	47.4%	83.9%	46.1%
Bi-GNN	88.0%	44.6%	81.5%	41.3%
DiffPool-S	87.6%	38.2%	77.1%	33.6%
O-GNN	87.4%	25.8%	64.2%	19.2%
Bi-HGNN	**90.5%**	**59.1%**	**90.2%**	**63.2%**

Table 3: Performance comparison of *Bi-HGNN* to the baseline methods on the E-Commerce dataset.

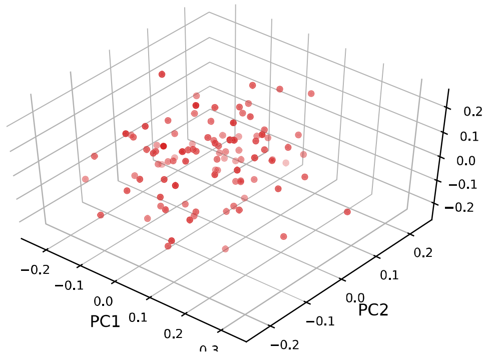

Figure 5: Visualization of the first three Principal Components of the *community space embedding*, which is centered and normalized. The dimension of the *community space embedding* is 256 and the number of community is 100.

The t-Distributed Stochastic Neighbor Embedding (t-SNE) [van der Maaten and Hinton, 2008] of the trained community space embedding is shown in Figure 4. In Figure 5, we also visualized the first three principal components [Rokhlin *et al.*, 2010] of the trained community space embedding. As shown in Figure 5, most of the communities scattered nicely in the space, but some of the communities are not spaced far enough to distinguishable. Developing systematic methods to properly choosing the number of communities is the next step of this research.

5 Conclusion

GNNs have achieved state-of-the-art results in various graph representation tasks. However, most existing GNNs are inherently flat and unable to capture the hierarchical structure of graphs. We propose *Bi-HGNN* a recommender system for the E-Commerce platform that effectively utilizes the users' community-level generalization as well as their individualized preferences. Our proposed method is in clear contrast with other hierarchical GNNs methods that ignore the user-specific information that is not captured by the community-level generalization. We evaluated our proposed on a world-leading E-Commerce platform with satisfactory performance.

References

[Abu-El-Haija *et al.*, 2018] Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alex Alemi. Watch your step: Learning node embeddings via graph attention. In *Neural Information Processing Systems*, 2018.

[Bojchevski and Günnemann, 2018] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018.

[Chen *et al.*, 2018] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. Harp: Hierarchical representation learning for networks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, 2018.

[Chen *et al.*, 2019] Zhengdao Chen, Lisha Li, and Joan Bruna. Supervised community detection with line graph neural networks. In *International Conference on Learning Representations*, 2019.

[Cheng *et al.*, 2016] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10. ACM, 2016.

[Ekstrand *et al.*, 2011] Michael D. Ekstrand, John T. Riedl, and Joseph A. Konstan. Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction*, 4(2):81–173, 2011.

[Golub and Reinsch, 1971] Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. In *Linear Algebra*, pages 134–151. Springer, 1971.

[Grbovic and Cheng, 2018] Mihajlo Grbovic and Haibin Cheng. Real-time personalization using embeddings for search ranking at airbnb. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18. ACM, 2018.

[Grbovic, 2017] Mihajlo Grbovic. Search ranking and personalization at airbnb. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 339–340, New York, NY, USA, 2017. ACM.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864, 2016.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[Moulton and Jiang, 2018] Ryan Moulton and Yunjiang Jiang. Maximally consistent sampling and the jaccard index of probability distributions. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 347–356. IEEE, 2018.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014.

[Ribeiro *et al.*, 2017] Leonardo F. R Ribeiro, Pedro H. P Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

[Rokhlin *et al.*, 2010] Vladimir Rokhlin, Arthur Szlam, and Mark Tygert. A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1100–1124, 2010.

[Tang *et al.*, 2015] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077, 2015.

[van der Maaten and Hinton, 2008] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[Ying *et al.*, 2018a] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2018.

[Ying *et al.*, 2018b] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pages 4805–4815, 2018.

[Zhang *et al.*, 2006] Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. Learning from incomplete ratings using non-negative matrix factorization. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 549–553. SIAM, 2006.

[Zhu *et al.*, 2018] Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. Learning tree-based deep model for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2018.

[Zitnik and Leskovec, 2017] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):190–198, 2017.