Detecting User Community in Sparse Domain via Cross-Graph Pairwise Learning

Zheng Gao
Indiana University Bloomington
gao27@indiana.edu

Hongsong Li
Alibaba Group
hongsong.lhs@alibaba-inc.com

Zhuoren Jiang
Zhejiang University
jiangzhuoren@zju.edu.cn

Xiaozhong Liu
Indiana University Bloomington
liu237@indiana.edu

ABSTRACT

Cyberspace hosts abundant interactions between users and different kinds of objects, and their relations are often encapsulated as bipartite graphs. Detecting user community in such heterogeneous graphs is an essential task to uncover user information needs and to further enhance recommendation performance. While several main cyber domains carrying high-quality graphs, unfortunately, most others can be quite sparse. However, as users may appear in multiple domains (graphs), their high-quality activities in the main domains can supply community detection in the sparse ones, e.g., user behaviors on Google can help thousands of applications to locate his/her local community when s/he uses Google ID to login those applications. In this paper, our model, Pairwise Cross-graph Community Detection (PCCD), is proposed to cope with the sparse graph problem by involving external graph knowledge to learn user pairwise community closeness instead of detecting direct communities. Particularly in our model, to avoid taking excessive propagated information, a two-level filtering module is utilized to select the most informative connections through both community and node level filters. Subsequently, a Community Recurrent Unit (CRU) is designed to estimate pairwise user community closeness. Extensive experiments on two real-world graph datasets validate our model against several strong alternatives. Supplementary experiments also validate its robustness on graphs with varied sparsity scales.

CCS CONCEPTS

• **Information systems** → *Information retrieval*; • **Unsupervised learning** → *Cluster analysis*; • **Machine learning approaches** → Neural networks.

KEYWORDS

cross graph, community detection, pairwise learning

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401055

ACM Reference Format:
Zheng Gao, Hongsong Li, Zhuoren Jiang, and Xiaozhong Liu. 2020. Detecting User Community in Sparse Domain via Cross-Graph Pairwise Learning. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401055

1 INTRODUCTION

Community detection is an essential task for cyberspace mining, which has been successfully employed to explore users' resemblance for retrieval/recommendation enhancement and user behavior analysis. Taking social media and e-commerce as examples, the complex, and often heterogeneous, relations among users and other objects, e.g., products, reviews, and messages, can be encapsulated as bipartite graphs, and the topology can help to synthesize and represent users with a coarser and broader view.

While a graph is well-connected, conventional methods, e.g., modularity-based approach [20], spectral approach [35], dynamic approach [11] and deep learning approach [34], are able to estimate the internal/external connectivity and generate high-quality communities directly on nodes [14].

For a vulnerable graph with sparse connectivity, however, prior community detection algorithms can hardly probe enough information to optimize the community structure. Unfortunately, in the real cyberspace this can be a very common problem, i.e., while a handful of giant players (like Google, Facebook and Amazon) maintaining high-quality graphs, thousands of apps are suffering from graph cold start problem. If many users are isolated because of data sparseness, we can hardly tell any community information.

In order to cope with this challenge, in this paper, we propose a novel research problem – Cross Graph Community Detection. The idea is based on the fact that an increasing number of small apps are utilizing the user identity information inherited from giant providers, i.e., users can easily login a large number of new apps by using Facebook and Google ID. In such ecosystem, the main large graph can provide critical information to enlighten the community detection on many small sparse graphs. Figure 1 depicts an example, where the Cooking or Cosmetic Apps inherits important topological information from the main Amazon graph for their enhanced community detection.

Note that, while the small sparse graphs can engage with a local field (like cooking or cosmetic in the example), the main graph can be quite comprehensive and noisy. As Figure 1 shows, not

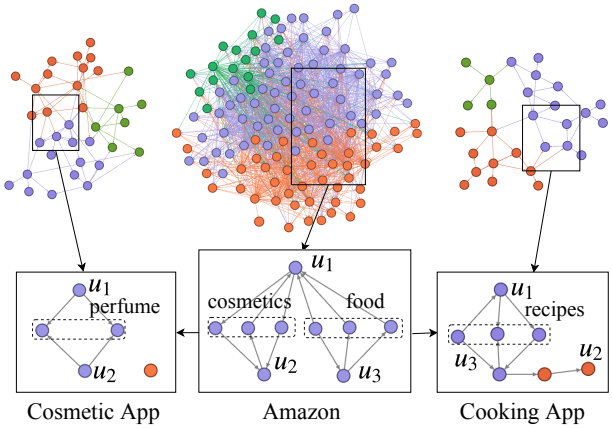


Figure 1: The Amazon graph is a main shopping graph while the Cosmetic App and the Cooking App are two small sparse graphs. Those graphs share mutual users in which three of them are selected to demonstrate their behaviors. Node colors indicate their related communities.

all the connections in Amazon (shopping graph) can be equally important for the two candidate app graphs. Three mutual users are selected where u_1 and u_2 mainly share similar shopping interests on cosmetics and u_1 and u_3 mainly share similar shopping interests on food products in Amazon. Then, with deliberate propagation from main graph, in the Cosmetic graph, u_1 and u_2 have a better chance to be grouped together, while u_1 and u_3 are more likely to be assigned the same community ID in the Cooking graph. Therefore, the proposed model should be able to differentiate various kinds of information from the main graph for each candidate sparse graph to enhance its local community detection performance.

As another challenge, small sparse graphs often suffer from training data insufficiency, e.g., the limited connections in these graphs can hardly tell the community residency information. In this paper, we employed a novel data augmentation approach - cross graph pairwise learning. Given a candidate user and an associated user triplet, the proposed model can detection the community closeness superiority by leveraging main graph and the sparse graph simultaneously. Moreover, the proposed pairwise learning method can reduce noisy information by taking care of graph local structure. Theoretically, we can offer at most $O(N^3)$ user triplets to learn graph community structure while conventional community detection methods by default can only be applied on $O(N)$ users (N is the number of users in the sparse graph).

Inspired by aforementioned discussion, we propose an innovative *Pairwise Cross-graph Community Detection* (PCCD) model for enhanced sparse graph user community detection. Specifically, given user u_i and its associated triplet $\langle u_i, u_j, u_k \rangle$, we aim to predict their pairwise community relationship, e.g., compared with user u_k, user u_j should have closer, similar or farther community closeness to user u_i. The contribution of this paper is fourfold:

- We propose a novel problem - Cross Graph Community Detection, which can be critical for thousands of small businesses or services if they enable external user account login.
- Unlike conventional community detection methods, we explore community structure from a pairwise viewpoint. In this way, we can efficiently deal with bipartite graph information and solve cold-start problem for users with no behaviors in sparse graphs.
- The proposed model is trained in an end-to-end manner where a two-level filtering module locates the most relevant information to propagate between graphs. A Community Recurrent Unit (CRU) subsequently learns user community distribution from the filtered information.
- Extensive experiments on two real-world cross-graph datasets validate our model superiority. We also evaluate our model robustness on graphs with varied sparsity scales and many other supplementary studies.

Reproducibility: In order to motivate other scholars' further investigations, the code and dataset will be publicly available once published (the project URL is omitted for review reason).

2 METHOD

2.1 Task Overview

As aforementioned, conventional methods suffer from graph sparseness problem. In this study, we propose a Pairwise Cross-graph Community Detection (PCCD) model that particularly aims at detecting user pairwise community closeness in sparse graphs by involving cross-graph techniques.

Particularly, a well connected graph is called "main graph" in this paper, corresponding to the targeted sparse graph. In general, as users may visit multiple cyber domains within a short time period, these mutual users co-occurred in both the main and sparse graph are taken as the bridge to connect the two graphs. Therefore, the relevant information from the main graph can be propagated to the sparse graph to support its user community detection.

Specifically, our proposed model (showed in Figure 2) is trained on mutual user triplets to learn three types of pairwise community relationship including "*similar*", "*closer*" and "*farther*". The goal of this study can be reformulated as the following pairwise learning task: Given a sparse graph S and a main graph M where N^C denotes their mutual user collection, for a mutual user triplet $\langle u_i, u_j, u_k \rangle \in N^C$ (**Model Input**), we aim to predict the relationship of its pairwise community closeness in graph S (**Model Output**). In this paper, "*similar*" relationship means that u_j and u_k are either in the same community or different communities with u_i. "*closer*" relationship means that u_j and u_i are in the same community, while u_k and u_i are in different communities. "*farther*" relationship means that u_j and u_i are in different communities, while u_k and u_i are in the same community.

For a mutual user triplet, our proposed model detects communities firstly on separate graphs to obtain all user raw community representations (Section 2.2). Their propagative representations are learned through a community-level and a node-level filter (Section 2.3). Both representations are jointly utilized for predicting user community affiliation in each graph via a Community Recurrent Unit (Section 2.4). In the end, we integrate the community affiliation distributions to identify the pairwise community relationship of the triplet (Section 2.5). Particular training strategies are introduced in the end (Section 2.6).

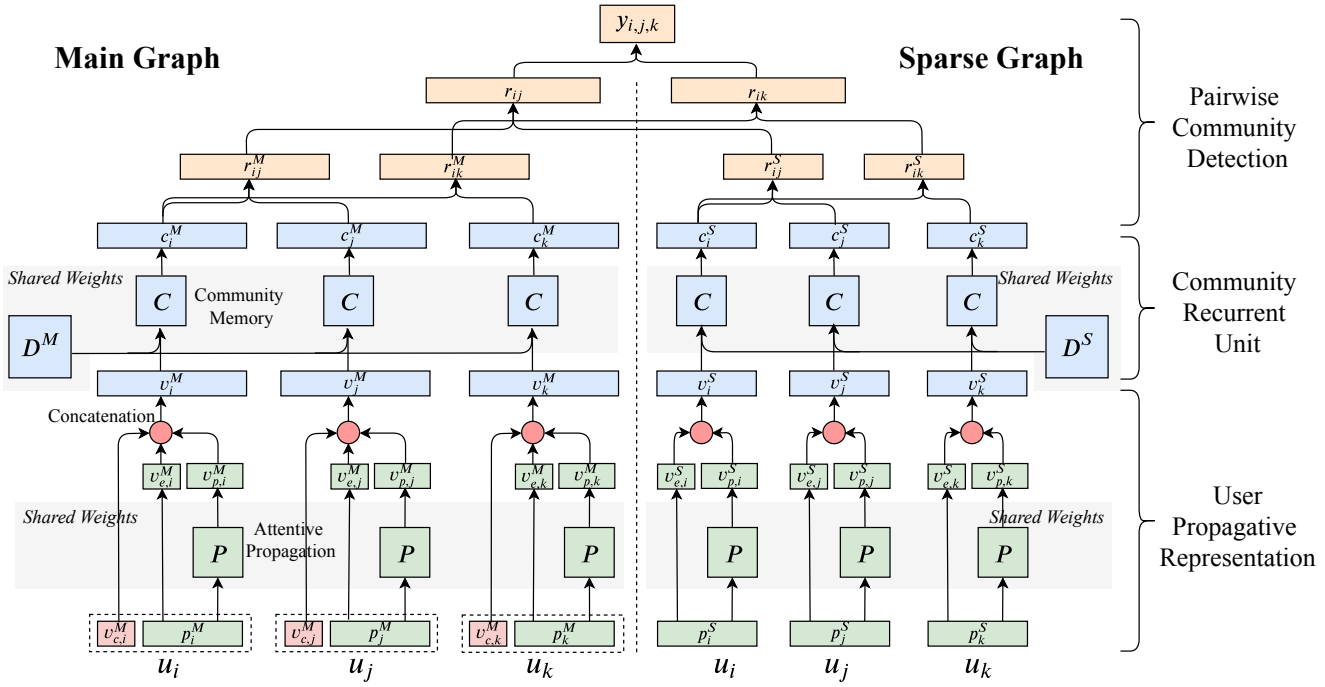


Figure 2: The overall architecture of our proposed PCCD model. It contains three major modules mentioned in the right part of the figure. Each training instance is a mutual user triplet $\langle u_i, u_j, u_k \rangle$. Compared with the sparse graph, the main graph involves raw user community as part of model input.

Note that even though our model is only trained on mutual users, it is still functional to detect pairwise community closeness on those users solely appeared in either the main or sparse graph. Further experiments in Section 3.5 will verify its effectiveness on different types of users.

2.2 Raw Community Representation

As the user behavior is enriched in the main graph M, detecting user communities in it would offer auxiliary community features to potentially enhance our model performance. Therefore, we decide to involve user communities in the main graph as a part of model input. The same information from the sparse graph is intentionally omitted because its sparse graph structure is not able to offer reliable community partition results.

Considering all possible community detection methods listed in Section 3.2, Infomap [24] is empirically selected to detect user communities in the main graph M. Infomap method simulates a random walker wandering on the graph for m steps and indexes his random walk path via a two-level codebook. Its final goal aims to generate a community partition with the minimum random walk description length, which is calculated as follows:

$$L(\pi) = \sum_{i}^{m} q^{i}_{\curvearrowright} H(\mathcal{Q}) + \sum_{i=1}^{m} p^{i}_{\circlearrowleft} H(\mathcal{P}^{i}) \tag{1}$$

where $L(\pi)$ is the description length for a random walker under current community partition π. $q^{i}_{\curvearrowright}$ and $p^{i}_{\circlearrowleft}$ are the jumping rates between and within the i_{th} community in each step. $H(\mathcal{Q})$ is the frequency-weighted average length of codewords in the global index codebook and $H(\mathcal{P}^{i})$ is frequency-weighted average length of codewords in the i_{th} community codebook.

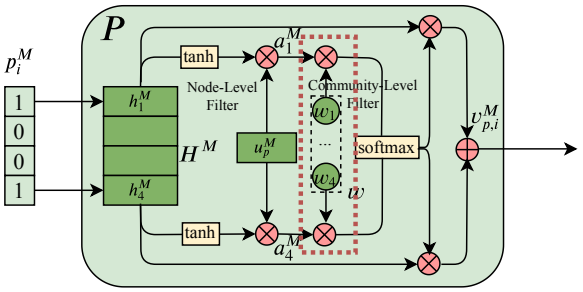


Figure 3: The two-level filter to support user propagative representation estimation in the main graph M. Sparse graph S does not contain the community-level filter (wrapped in the red rectangle).

In the end, given a user u_i, we obtain its one-hot community representation $v^{M}_{c,i}$ in the main graph M, which is regarded as part of user representations for the model input.

2.3 User Propagative Representation

To better convey the mutual user insights in both main and sparse graph, their representations are learned from both direct transformation and weighted information propagation. The optimization processes are similar in the main and sparse graph. In the beginning,

a user u_i can be represented as a multi-hot embedding p_i^M in the main graph M (left part in Figure 3) and p_i^S in the sparse graph S. Each dimension in the embedding refers to a unique object with which the user connects in the related graph.

The direct transformation from user multi-hot embedding learns a dense vector via one hidden layer so as to retrieve the compressed information from user connected objects directly, which is calculated as follows:

$$v_{e,i}^G = \tanh(W_e^G p_i^G + b_e^G) \tag{2}$$

where $G \in \{M, S\}$ denotes either the main or the sparse graph. W_e^G and b_e^G denote the weight matrix and bias respectively. $v_{e,i}^G$ is the learned direct transformation representation for user u_i.

On the other hand, from information propagation perspective, each object carries information, which can be propagated to its connected users as a type of representation. However, as such information are noisy, not all of them are equally important to users. Therefore, a filtering module is proposed to automatically select appropriate information to propagate from both community level and node level. The overall flow of the filtering module is illustrated in Figure 3.

Community-Level Filter: As the raw detected communities in the sparse graph is not reliable because of graph sparseness, we only apply the community-level filter in the main graph M. According to the raw community partition π calculated in Section 2.2, each object node n_i is assigned to a community $\pi(n_i)$. The community-level filter propagate object information in a coarser manner by considering its community importance weight $w_{\pi(n_i)}$. Throughout this filter, we aim to learn a weight vector w where each dimension of w denotes the importance score of a related community.

Node-Level Filter: Node-level filter assesses the propagated information with a fine resolution. In the main graph M (a user-object bipartite graph), we aim to learn object representations $H^M = \{h_1^M, ..., h_n^M\}$ where h_n^M denotes the n_{th} object representation. Similarly, H^S denotes the object representations in the sparse graph S. The weight of the j_{th} object node decides the amount of its information propagating to its connected user nodes, which is calculated in an attentive means:

$$a_j^G = (u_p^G)^\intercal \tanh(W_p^G h_j^G + b_p^G) \tag{3}$$

where $G \in \{M, S\}$ denotes either the main or the sparse graph. u_p^G denotes the project vector in the related graph to calculate object attentive weights. W_p^G and b_p^G denote the corresponding weights and bias, respectively.

We combine the community-level and node-level weights together to quantify the information propagation in the main graph M. While only the node-level weights is considered in the sparse graph S. Normalized by the softmax$(\cdot)$ function, the related representation of user u_i in both graphs are calculated as follows:

$$\begin{aligned} v_{p,i}^M &= \sum_{n_j \in \mathcal{N}^M(u_i)} \text{softmax}(a_j^M w_{\pi(n_j)}) h_j^M \\ v_{p,i}^S &= \sum_{n_j \in \mathcal{N}^S(u_i)} \text{softmax}(a_j^S) h_j^S \end{aligned} \tag{4}$$

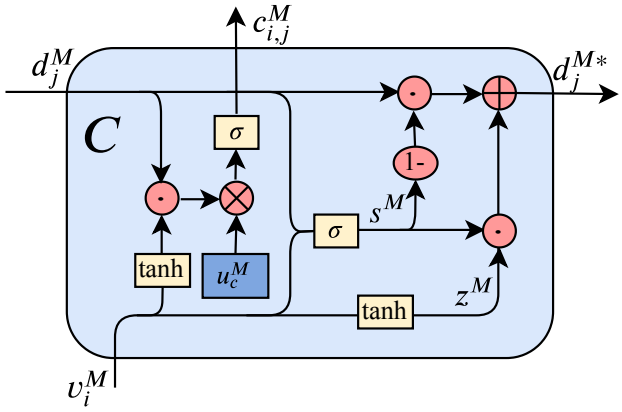


Figure 4: The flow of Community Recurrent Unit in the main graph M. The left part refers to the community affiliation gate and the right part is the community update gate.

Both $v_{p,i}^M$ and $v_{p,i}^S$ are the weighted sum of the neighbour object representations. $\mathcal{N}^S(n_i)$ and $\mathcal{N}^M(n_i)$ denote user connected objects in both graphs.

Finally, given a user u_i, its raw community representation, direct transformation representation and two-level filtered representation construct its propagative representation in the main graph M. While its direct transformation representation and node-level filtered representation construct its propagative representation in the sparse graph S. To avoid gradient vanishing, a batch_norm$(\cdot)$ function is applied on top of the concatenated representations in both graph:

$$\begin{aligned} v_i^M &= \text{batch\_norm}([v_{c,i}^M, v_{e,i}^M, v_{p,i}^M]) \\ v_i^S &= \text{batch\_norm}([v_{e,i}^S, v_{p,i}^S]) \end{aligned} \tag{5}$$

2.4 Community Recurrent Unit

After previous steps, each user of the mutual user triplet $\langle u_i, u_j, u_k \rangle$ is associated with a propagative representation. In this section, its corresponding community affiliation scores c_i^G are further calculated in related graphs $G \in \{M, S\}$ through a designed Community Recurrent Unit (CRU), which is showed in Figure 4. The CRU contains an affiliation gate to calculate the user affiliation score for each community and an update gate to update community self representations. Within this unit, a community memory $D^G = \{d_1^G, ..., d_K^G\}$ is designated to store K community representations. Particularly, community representation is required to be with the same dimension as user propagative representation in both graphs.

Community Affiliation Gate: The affiliation gate helps to generate the affiliation score of a user u_i towards each community in both graphs, which forms a K-dimensional vector $c_i^G = \{c_{i,1}^G, ..., c_{i,K}^G\}$ where $G \in \{M, S\}$. The user u_i's affiliation score $c_{i,j}^G$ towards the j_{th} community in the graph G is calculated as follows:

$$c_{i,j}^G = \sigma((u_c^G)^\intercal (\tanh(v_i^G) * d_j^G)) \tag{6}$$

The dot product between transformed user propagative representation $\tanh(v_i^G)$ and the j_{th} community representation d_j^G indicates

their potential correlation, which further turns to a scalar affiliation score $c_{i,j}^G$ between 0 to 1 with the help of a projection vector u_c^G and normalization function $\sigma(\cdot)$.

Community Update Gate: When calculating u_i's community affiliation, its information can help to update community representations in return. The updated representation is relied on both previous step community representation and current user propagative representation. Therefore, to better embrace the two representations, we use a delicate RNN variant, where the update process is calculated as follows:

$$\begin{aligned} s^G &= \sigma(W_s^G[v_i^G, d_j^G] + b_s^G) \\ z^G &= \tanh(W_z^G v_i^G + b_z^G) \\ d_j^{G*} &= (1 - s^G) * d_j^G + s^G * z^G \end{aligned} \tag{7}$$

where $G \in \{M, S\}$. s^G denotes the update rate and z^G is transformed user information to be updated in current community. d_j^{G*} denotes the updated representation of the j_{th} community in graph G, which is the weighted sum on previous community and current user representations. W_s^G and W_z^G, denote related weight matrices, while b_s^G and b_z^G, denote related biases.

Community Constraint: To obtain the communities with less overlapping information in graph $G \in \{M, S\}$, the community representations ideally should be independent with each other. In our model, cosine similarity $\cos(\cdot)$ is taken as the criteria to measure the community similarity where higher score indicates stronger correlation. The averaged cosine similarities of all possible community representation pairs is calculated as a community loss to minimize:

$$\mathcal{L}_c^G = \frac{1}{2K^2} \sum_{i,j} \cos(d_i^G, d_j^G) \tag{8}$$

where K is the number of communities in each graph.

2.5 Pairwise Community Detection

Similar to RankNet [3], given a mutual user triplet $\langle u_i, u_j, u_k \rangle$, as three types of pairwise label are considered including "*closer*", "*similar*" and "*farther*", label $y_{i,j,k}$ is calculated as follows:

$$y_{i,j,k} = \frac{1}{2}(1 + S_{jk}) \tag{9}$$

In terms of the community closeness for u_i, if u_j is closer than u_k, $S_{jk} = 1$; if u_j is farther than u_k, $S_{jk} = -1$; if u_j and u_k are similar, $S_{jk} = 0$. In this way, we convert the pairwise ranking task to a regression task.

To optimize this task, first of all, we calculate the correlation representation r_{ij}^G between u_j and u_i in single graph $G \in \{M, S\}$:

$$r_{ij}^G = W_r^G(c_i^G - c_j^G) + b_r^G \tag{10}$$

where W_r^G and b_r^G are related weight and bias, respectively.

To construct the information from both graphs, we concatenate the correlation representations from both graphs:

$$r_{ij} = \tanh([r_{ij}^S, r_{ij}^M]) \tag{11}$$

Similarly, the cross-graph correlation representation between u_i and u_k is calculated as r_{ik}.

After that, we predict the community relationship between u_j and u_k towards u_i as follows:

$$\hat{y}_{i,j,k} = \sigma(W_o(r_{ij} - r_{ik}) + b_o) \tag{12}$$

where W_o and b_o are the related weight and bias. $\sigma(\cdot)$ denotes the sigmoid activation function. In the end, the final loss $\mathcal{L}_{total}$ is the weighted sum of the optimization loss calculated via cross entropy and the community constraint losses where $\mathcal{L}_c^S$ is for the sparse graph and $\mathcal{L}_c^M$ is for the main graph:

$$\mathcal{L}_{total} = \underbrace{-y_{i,j,k} log \hat{y}_{i,j,k} - (1 - y_{i,j,k}) log(1 - \hat{y}_{i,j,k})}_{optimization} + \underbrace{\alpha(\mathcal{L}_c^S + \mathcal{L}_c^M)}_{constraint} \tag{13}$$

2.6 Training Strategy

Although our model is trained solely on mutual user triplets, it is also able to detect pairwise community closeness among users only appeared in either sparse graph or main graph. Two training tricks are particularly employed in our model to improve model robustness, including shared weights and masked training.

First, as our model should be insensitive to the sequence of input user triplet, the weight matrices, biases and project vectors in both Section 2.3 and Section 2.4 should be shared among the three users. Moreover, in Section 2.4, the community memory D^G where $G \in \{M, S\}$ is updated by taking the average of the three updated community representations (calculated in Eq. 7) from input triplets.

Second, to alleviate the negative effects of excessive connections in main graph M, we employ a masked training strategy by randomly remove a small ratio ρ of connected objects from users in the main graph during each training batch. $\rho = 0$ means users remain their original multi-hot embeddings while $\rho = 1$ means users lose all their connections on objects.

3 EXPERIMENT

3.1 Dataset

As one of the largest e-commerce ecosystems, Alibaba owns multiple online businesses, where users could try different services via the same login ID. For this experiment, we employ three services from Alibaba including an online shopping website, a digital news portal and a cooking recipe website. The online shopping website is regarded as the main graph because it is Alibaba core service having the largest number of active users. By contrast, the other two services are regarded as sparse graphs. In the online shopping website, each object refers to a product for sale, and links are user purchase behaviors on products. In the digital news portal, each object refers to a news article, and links are user reads on news. In the cooking recipe website, each object refers to a cooking recipe, and links mean that users follow recipe instructions.

To prove our model robustness with respect to the graph size, two cross-graph datasets are constructed in different scales, which are showed in Table 1. The SH-NE dataset is the cross-graph dataset constructed by the shopping graph and news graph, which is ten times larger than the SH-CO dataset constructed by the shopping graph and recipe graph. Users are categorized into three different types including *MU* (mutual users appeared in both graphs), *MO*

(users only appeared in the main graph), and *SO* (users only appeared in the sparse graph). Note that, only mutual user triplets are used for training.

Dataset	#MU	Main Graph			Sparse Graph		
		#MO	#Object	#Link	#SO	#Object	#Link
SH-NE	105,702	392,549	135,320	17,352,482	26,133	9,377	881,721
SH-CO	1,029	1,543	23,881	1,161,293	719	3,739	147,266

Table 1: Statistics of Two cross-graph datasets.

Both datasets are built by taking a week of user behaviors from 09/20/2018 to 09/26/2018. In this paper, for two sparse graphs, we aim to predict user pairwise community labels (calculated from enriched seven-day user behaviors) using sparse user behaviors (the first four-day user behaviors, arbitrarily defined in this paper).

To construct our ground truth data, we run all baseline models (will be introduced in Section 3.2) on both seven-day sparse graphs which are assumed to contain sufficient user-product connections during this relatively long time period. Only mutual user communities agreed by all six baselines (except random model) are kept, from which mutual user triplets are subsequently selected as we assume the commonly-agreed communities are reliable. In this paper, we define three types to label pairwise community closeness for a mutual user triplet, including "*closer*", "*similar*", and "*farther*" (detailed definition is in Section 2.1). Equal number of triplets are randomly selected for each label type. In total, there are 207,291 triplets in SH-NE dataset and 20,313 triplets in SH-CO dataset for testing propose.

Training data generation is the same as ground truth data construction except using first four-day user behaviors. In total, there are 2,072,905 triplets in SH-NE dataset and 203,123 triplets in SH-CO dataset, from which 90% of the triplets are used for training and 10% for validation.

In both training and testing process, our model input is user triplets with first four-day behavior information. Their difference is that the pseudo labels for training are generated from four-day user behaviors, while the ground truth labels for testing are generated from whole seven-day user behaviors in sparse graphs.

3.2 Baselines and Settings

As there is neither reliable nor open-sourced studies on pairwise cross-graph community detection, we have no direct comparative models. In this paper, we select one random model and six compromised but state-of-art baselines which are originally designed for whole graph community detection. In all seven baselines, after user communities are detected, we calculate user triplet labels via Eq. 9. As all baselines are trained on the sparse graph with first four-day user behavior and the main graph, they bring no information loss compared with our model. Here is the details of how these baselines calculate user communities:

- **random**: users are randomly assigned to a given number of communities.
- **Infomap** [24]: A random walk based method to detect user communities by minimizing description length of walk paths[1].
- **DNR** [39]: A novel nonlinear reconstruction method to detect communities by adopting deep neural networks on pairwise constraints among graph nodes[2].
- **LSH** [5]: A random walk based method for community detection in large-scale graphs[3].
- **node2vec** [18]: A skip-gram model to generate node emebeddings based on guided random walks on graphs. And K-means method subsequently calculates communities from the generated embeddings[4].
- **BigClam** [38]: A non-negative matrix factorization method to detect overlapping communities in large scale graphs[4].
- **WMW** [4]: A fast heuristic method for hierarchical community detection inspired by agglomerative hierarchical clustering[5].

The result is reported with best hyper-parameters from grid search. Empirically, a mini-batch (size = 200) Adam optimizer is chosen to train our model with learning rate as 0.01. Community number on both graphs is $K = 50$ (Eq. 8). Community constraint weight is $\alpha = 0.1$ (Eq. 13). Masked training rate in the main graph is $\rho = 0.05$. The model is trained for 1000 epochs in order to get a converged result.

Model performance is evaluated from both classification and retrieval viewpoints. From classification aspect, Accuracy (ACC), F1-macro (F1) and Matthews Correlation Coefficient (MCC) are reported metrics. From retrieval aspect, Mean Reciprocal Rank (MRR), Normalized Discounted Cumulative Gain (NDCG) and Mean Average Precision (MAP) are reported metrics. All of them are fundamental metrics widely used for model evaluation.

3.3 Performance Comparison

All model performance are evaluated from both classification and retrieval viewpoints. Each baseline method is run on three graphs including the main graph, the sparse graph, and their combined graph. While our model can only be tested by utilizing both graphs given the proposed model structure. Table 2 shows the overall model performance under three graphs. Particularly, we run our model ten times and report the average performance result. To verify our model's superiority, we calculate the performance differences between our model and the best baseline on each metric for all the runs, and apply a pairwise t-test to check whether the performance difference is significant.

Our model output is a predicted score in the range from 0 to 1. Its associated label is assigned to be one of "*closer*", "*similar*" or "*farther*" based on which third the predicted score lies in (associated with label definition in Section 2.5). For example, a user triplet $\langle u_i, u_j, u_k \rangle$ with score 0.3 will be labelled as "*farther*", which means compared with u_k, u_j is has a farther community closeness to u_i. After that, from classification perspective, ACC, F1 and MCC scores are calculated by comparing predicted labels with the ground truth labels. While from retrieval perspective, MRR, NDCG and MAP are calculated for the pairwise ranking within user triplets.

[1] https://github.com/mapequation/infomap

[2] http://yangliang.github.io/code/

[3] https://github.com/melifluos/LSH-community-detection

[4] https://github.com/snap-stanford/snap/

[5] https://github.com/eduarc/WMW

Graph	Model	SH-NE Dataset						SH-CO Dataset					
		ACC	F1	MCC	MRR	NDCG	MAP	ACC	F1	MCC	MRR	NDCG	MAP
Main	random	0.3324	0.1677	-0.0001	0.7719	0.8373	0.7299	0.3302	0.1700	0.0027	0.7842	0.8444	0.7401
	Infomap	0.3700	0.2731	0.1201	0.8334	0.8421	0.7299	0.3811	0.3723	0.1000	0.7888	0.8419	0.7430
	DNR	0.4210	0.4022	0.1789	0.8000	0.8366	0.7473	0.5194	0.5122	0.2712	0.8099	0.8773	0.7592
	LSH	0.3798	0.3813	0.0888	0.8002	0.8433	0.7328	0.4000	0.4123	0.1277	0.7812	0.8329	0.7570
	node2vec	0.4888	0.5014	0.2521	0.8229	0.8744	0.7832	0.5959	0.5832	0.3895	0.8422	0.8936	0.8222
	BigClam	0.5555	0.5399	0.3349	0.8421	0.8890	0.8024	0.5301	0.5334	0.3290	0.8335	0.8573	0.8005
	WMW	0.6032	0.6158	0.4111	0.8632	0.8988	0.8256	0.6489	0.6661	0.5001	0.8789	0.9032	0.8499
Sparse	random	0.3333	0.1785	0.0003	0.7815	0.8438	0.7419	0.3301	0.1720	0.0030	0.7801	0.8400	0.7338
	Infomap	0.3455	0.1929	0.0031	0.7843	0.8399	0.7437	0.3676	0.3211	0.0422	0.7905	0.8446	0.7401
	DNR	0.4433	0.4407	0.2020	0.8313	0.8678	0.7742	0.5273	0.5289	0.2787	0.8293	0.8765	0.7980
	LSH	0.3889	0.3917	0.0942	0.8008	0.8528	0.7343	0.4093	0.4177	0.1138	0.7833	0.8438	0.7404
	node2vec	0.4849	0.4958	0.2467	0.8219	0.8664	0.7774	0.5732	0.5746	0.3898	0.8412	0.8823	0.8146
	BigClam	0.5277	0.5355	0.3332	0.8441	0.8750	0.8005	0.5282	0.5280	0.3339	0.8298	0.8399	0.8033
	WMW	0.5814	0.5900	0.3988	0.8666	0.9001	0.8210	0.5911	0.6212	0.4347	0.8599	0.8890	0.8133
Main + Sparse	random	0.3337	0.1788	-0.0008	0.7815	0.8387	0.7387	0.3291	0.1699	0.0027	0.7739	0.8331	0.7310
	Infomap	0.3625	0.2567	0.0908	0.7864	0.8423	0.7437	0.3930	0.3988	0.0916	0.7951	0.8488	0.7528
	DNR	0.4383	0.4204	0.1817	0.8101	0.8599	0.7686	0.5273	0.5322	0.2957	0.8391	0.8812	0.8000
	LSH	0.3992	0.4052	0.0993	0.8020	0.8538	0.7599	0.4180	0.4237	0.1309	0.8021	0.8539	0.7600
	node2vec	0.5072	0.5124	0.2611	0.8339	0.8774	0.7943	0.6015	0.6055	0.4045	0.8677	0.9023	0.8324
	BigClam	0.5580	0.5626	0.3412	0.8511	0.8901	0.8135	0.5398	0.5383	0.3389	0.8475	0.8874	0.8095
	WMW	0.6168	0.6205	0.4281	0.8702	0.9042	0.8353	0.6682	0.6711	0.5041	0.8879	0.9173	0.8561
	PCCD	**0.6524***	**0.6551***	**0.4808***	**0.8802***	**0.9116***	**0.8470***	**0.7740***	**0.7758***	**0.6627***	**0.9244***	**0.9442***	**0.9005***

Table 2: All model performances for three different graph combinations in two datasets. Symbol '*' highlights the cases where our model significantly beats the best baseline with $p < 0.01$.

Table 2 shows that almost all baseline performances in the combined graph achieve the best among three types of graphs. And performances in the main graph are always better than in the sparse graph. It demonstrates that main graph holds beneficial information to reveal user potential communities in sparse graphs. The random baseline is with similar performance among all three graphs, which makes sense because the random community partition is regardless of graph structures. It can be used as the default model to compare with in each graph. For the rest six baselines, random walk based baselines (Infomap and LSH) do not perform well in all three graphs. Their evaluation results are just a bit better than random model performance. Even though solely running Infomap does not perform well, it is the most supportive model for Section 2.2 Raw Community Detection (related empirical experiment comparisons are omitted as it is not the major model contribution). WMW model performs the best among all baselines. Actually, its results are always the best in three graphs of both datasets. However, by taking the pairwise t-test, it statistically proves that our model still significantly outperforms the WMW model in terms of all metrics.

It is also interesting to see that the model evaluation results are consistent in all datasets. For example, DNR model always performs better than LSH model. The only exception is that BigClam performs better than node2vec in SH-NE dataset, but worse in SH-CO dataset.

3.4 Ablation Study

In this section, we aim to explore whether all components of our proposed model have positive effect on final model performance and which component has the largest impact. There are six components that can be disassembled from the main model, including Raw Community Representation in the main graph (RCR), Direct Transformation Representation (DTR), Node-level filter (NF), Community-level filter (CF), Community Constraint (CC) and Masked Training (MT). We iteratively remove each component while keep the rest firmed to demonstrate their performance difference compared to the original model. All comparison results are showed in Table 3.

Among the six components, if we remove node-level filter, the performance drop dramatically, even worse than some of baseline models. It indicates that node-level information are excessive and noisy. Without appropriate filtering on the propagated information, it would vitally pollute the quality of user representations. Similarly, community-level filter also has a huge impact on model performance, meaning filtering propagated information from a coarser view also has a positive effect on learning user representations. By contrast, the two extra representations (RCR and DTR) do not have strong influence in our model. Having these information can slightly improve model performance as they still involve extra information. However, as one-hot embeddings, the amount of new information that RCR can offer is very limited. Similarly, DTR is only one-layer transformation on multi-hot embeddings. The information it can offer is also limited. By adding extra constraint, CC also has a little positive effect to enhance model performance. But as we always set its weight with a small value in training process, its influence is also indifferent. MT has the least impact on model performance as randomly masking user behaviors in the main graph

Dataset	Model	ACC	F1	MCC	MRR	NDCG	MAP
SH-NE	– RCR	-2.61	-2.51	-3.76	-0.73	-0.54	-0.96
	– DTR	-1.32	-1.49	-2.96	-1.01	-0.14	-0.57
	– NF	-17.31	-17.01	-25.92	-5.35	-3.88	-6.05
	– CF	-7.94	-8.48	-11.71	-2.61	-1.92	-3.01
	– CC	-2.15	-2.05	-3.08	-0.49	-0.35	-0.65
	– MT	-2.43	-2.55	-4.82	-0.78	-0.82	-1.24
SH-CO	– RCR	-2.13	-2.45	-3.07	-1.65	-0.58	-0.79
	– DTR	-3.45	-2.89	-1.95	-2.22	-0.43	-0.88
	– NF	-26.13	-25.83	-39.97	-9.81	-7.50	-19.35
	– CF	-21.12	-5.53	-8.45	-2.03	-1.50	-2.50
	– CC	-4.32	-4.34	-6.57	-1.56	-1.24	-1.91
	– MT	-1.32	-1.24	-2.66	-0.33	-0.21	-0.45

Table 3: Performance differences on all evaluation metrics between each ablated model and the original model. "-" refers to remove related component from our model. Results are scaled in percentage (%).

has an overlapped effect with attentive weights calculated from the node-level filter in Section 2.3.

3.5 User-Type Analysis

Dataset	User	ACC	F1	MCC	MRR	NDCG	MAP
SH-NE	MU	0.7132	0.7133	0.6012	0.9112	0.9324	0.8704
	MO	0.6356	0.6477	0.4780	0.8739	0.9002	0.8418
	SO	0.6009	0.6117	0.4324	0.8600	0.8988	0.8393
SH-CO	MU	0.7852	0.7780	0.6724	0.9300	0.9474	0.9015
	MO	0.7334	0.7565	0.6601	0.9012	0.9400	0.8874
	SO	0.6823	0.6915	0.6844	0.8990	0.9222	0.8789

Table 4: Three types of user performance in two datasets.

As aforementioned in Section 3.1, there are three types of users in our cross-graph datasets including *MU* (mutual users appeared in both graphs), *MO* (users only appeared in the main graph), and *SO* (users only appeared in the sparse graph). Unlike our training data which is constructed only with mutual users, our testing data can be with all types of users. To examine our model performance on each type of users, each single-type user user triplets are screened with five thousand instances, i.e., an eligible user triplet only contains three *MO* type users. In fact, the performance on *MO* user triplets reflects the capability of our model to solve cold-start problem as these users have no behaviors in the sparse graphs. The result showed in Table 4 demonstrates that the label of *MU* triplets can be best identified, which makes sense as the information of mutual users come from both graphs. While performance results on *MO* are better than *SO* user triplets, which might because that users are likely to have more enriched behaviors in the main graph compared with the sparse graph. Even that, the performance of our model on *SO* user triplets is still better than most of baselines running on the combined graph.

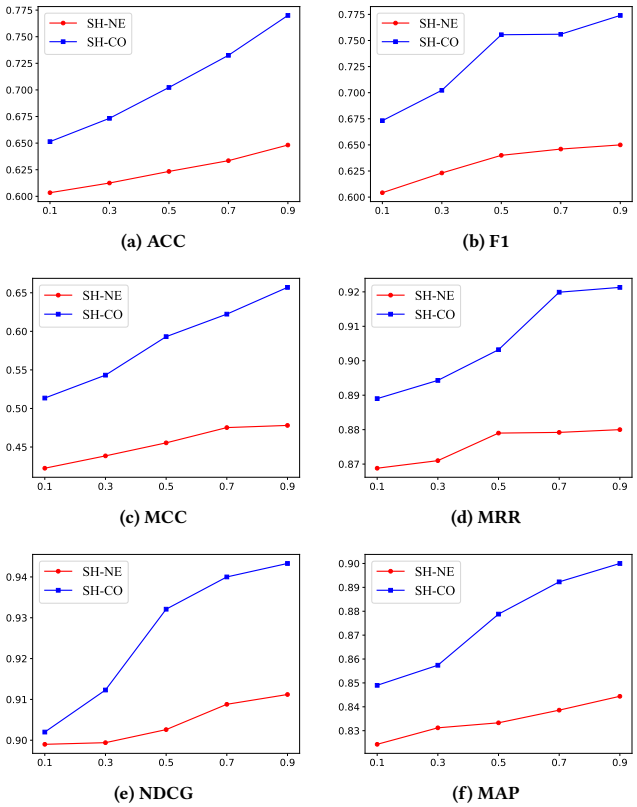


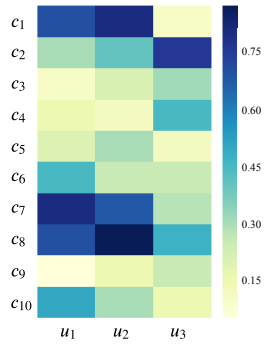
Figure 5: Our model performance on all metrics under different graph sparsity scales. X-axis is the ratio of links preserved in the sparse graph. Y-axis is related metric score.

3.6 Graph Sparsity Influence

To explore our model robustness for solving cold-start problem on sparse graphs, we randomly keep δ ratio of the total links in the sparse graph, where $\delta = \{0.1, 0.3, 0.5, 0.7, 0.9\}$. Our model is trained on the whole main graph and each truncated sparse graph. The reported evaluation metrics under all sparse graphs with varied scales are visualized in Figure 5. All metrics in both SH-NE and SH-CO datasets are in a rising trend with more links preserved in related sparse graphs. Compared with SH-NE dataset, SH-CO can benefit more from the sparse graph as it has a larger increase in all reported metrics.

With more links are preserved, all evaluation metrics first grows rapidly. While the increasing speed slows down when 70%-90% links are preserved. This phenomenon can be seen in almost all plots in Figure 5. It can be explained by the law of diminishing marginal utility, meaning that the marginal utility to bring new information derived from each additional link is declined.

Although classification related metrics significantly increase with more links preserved (i.e., F1 score in SH-CO dataset increases 10% when involving sparse graph), the retrieval related metrics (MRR, NDCG and MAP) do not change much in the mean time. One possible reason is that the information from the main shopping graph already contains enough information for the pairwise ranking



User	Keywords
u_1	maxi dress, long sleeve, pullover hoodie, oversized sweatshirt, jeans
u_2	hoodie, linen blazer, leather jacket, sportswear, legging short
u_3	window shades, light, kitchen countertops, toilet, ceiling fan

Figure 6: The left part shows ten community affiliation scores of three selected users in the shopping graph. The right part shows the keywords of their purchased products.

in user triplets. For example, without considering sparse graph information, our model has already achieved high MRR score as 0.87 in SH-NE dataset and 0.89 in SH-CO dataset, which is already better than most baseline results running on the combined graph.

3.7 Case Study

In order to testify whether our proposed Community Recurrent Unit is able to accurately calculate user affiliation scores in each community, we choose three users (labelled from user u_1 to user u_3) and calculate their affiliation scores of ten selected communities (labelled from community c_1 to community c_{10}) in the main shopping graph. The detailed result is visualized in Figure 6. In the left part of the figure, darker color indicates higher affiliation scores in the range from 0 to 1. Besides, in the shopping graph, we also extract all products purchased by each user, and manually select the most representative ones summarized as keywords in a table, which is demonstrated in the right part of Figure 6.

From the right-part table, u_1 and u_2 share similar shopping interests in clothing products. u_2 is more like a girl fond of sports. And u_1 more tends to be a fashion girl. While the shopping interests of u_3 is much far away from the other users. All purchased products by u_3 is furnishing stuffs. It seems s/he is decorating her/his living space. Referring to the left part of Figure 6, the affiliation distribution among ten communities between u_1 and u_2 are very similar. They both have a high weight in community c_1, c_7 and c_8. While the community distribution of u_3 is totally different. u_3 is heavily affiliated with community c_2. The results of our calculated community affiliation distribution is consistent with actual user behaviors, which indicates our Community Recurrent Unit (CRU) is functional well to reflect user community information.

4 RELATED WORKS

Exploring community structure in graphs has long been a fundamental research task in network science [36]. It aims to group densely connected nodes in the same community while setting weakly connected nodes apart. Modularity-based approaches explore the community structure with the largest predefined modularity score. Spectral approaches either extract the main components from graph Laplacians or cut the whole graph into several disjoint chunks. Dynamic approaches mostly detect communities from random walks by maximizing its probability or minimizing the walk path description length [2]. Recently, a surge of deep learning techniques are established to either learn node embeddings [15, 23] for spectral clustering, or propose novel modules to detect communities in an end-to-end fashion [30]. Some other approaches also focus on special scenarios, i.e. personalized community detection [16, 17] or community-based information diffusion [27].

However, conventional approaches are solely dependent on the node connections in a single graph [14], which is vulnerable when the graph connectivity is sparse. To cope with this problem, [1] perturbs sparse graphs by adding additional weak edges to connect isolated components. [19, 31] both argue to calculate Grothendieck's inequality on sparse graphs. [25] adds a nonbacktracking walk on the directed edges of sparse graphs to enhance existing spectral method performances. Similarly, [10] proposes a refined spectral method by adding an optimal recovery rate. Beyond adding subtle components into existing methods, two other types of approach can directly address sparse graph community detection problem by cross-graph learning (involving auxiliary information from other graphs) and pairwise learning (predicting the pairwise community relationship between nodes).

By jointly training on multi-graphs, mutual pattern can be shared across graphs [21]. [9] uses a non-negative matrix factorization with a designed co-regularized penalty on multiple graphs to manipulate their communities simultaneously. [29] introduces a unified model to jointly maximize the empirical likelihood and preserves the geometric structure in multi-graphs. [6] calibrates domain-specific graph Laplacians into a unified kernel, which detects graph patterns in semi-supervised fashion. [37] introduces a matrix factorization approach on two bipartite graphs simultaneously to measure the similarity between their shared nodes. [28] operates multi-layer spectral convolutions on different graphs to learn node communities. [12] proposes a regularized spectral clustering method to perform an efficient partitioning on multi-graphs. [32] introduces two different solutions to detect communities on multi-graphs based on minimum description length and tensor based decomposition principles.

Pairwise community detection offers an alternative view to avoid graph sparsity. Unlike other semi-supervised approaches [13, 40] which just regard node pairwise relationships as extra constraints for whole graph community detection, the pairwise community relationship between nodes is the predicting target. However, as a subtle track of community detection, it is neglected and starts to attract research attention until recently. Among one of the few works, [33] calculates the pairwise regularized KL-divergence from few labeled nodes, which is utilized to exploit the pairwise community relationship between abundant unlabeled data via convolutional autoencoder. [13, 22] extracts pairwise constraints using a mutual K-nearest neighbors approach to detect node pairwise community relationships. [26] leverages a multi-task to detect graph community and rank node pairwise community relationship simultaneously. [8] considers the pairwise proximity among the nodes of the complex networks and unveils community structure based on that. [7] uses a well-defined pairwise node similarity measure to identifies user communities in e-commerce.

5 CONCLUSION

Internet ecosystem presents star-shaped topology in the recent years: giant service providers, like Facebook, Google and Amazon, provide easy login channels to thousands of sites/apps, and users don't need experience laborious account creation anymore. In this study, motivated by this phenomena, we proposed a novel cross-graph community detection problem, which aims to detect user communities in sparse graphs by leveraging cross-graph information and pairwise learning techniques. It can substantially benefit small businesses or services to identify user groups and understand their interests. Extensive experiments on two real datasets demonstrate the superiority of our model comparing with all baselines. The limitation of this study is that we need to empirically detect communities to generate pseudo labels for training in a separate step. In the future, this process needs to be either omitted or merged to learn in an end-to-end manner. We also plan to upgrade our model by involving more auxiliary information, e.g., user/product textual profiles. Besides, Current model only considers information propagation between two graphs. A followup work needs to allow it run simultaneously among multiple graphs with appropriate graph-level filters.

REFERENCES

[1] Arash A Amini, Aiyou Chen, Peter J Bickel, Elizaveta Levina, and others. 2013. Pseudo-likelihood methods for community detection in large sparse networks. *The Annals of Statistics* 41, 4 (2013), 2097–2122.

[2] Aditya Ballal, Willow Kion-Crosby, and Alexandre Morozov. 2020. Inference of Network Communities using Random Walks. *Bulletin of the American Physical Society* (2020).

[3] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11, 23-581 (2010), 81.

[4] Eduar Castrillo, Elizabeth León, and Jonatan Gómez. 2017. Fast Heuristic Algorithm for Multi-scale Hierarchical Community Detection. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. ACM, 982–989.

[5] Benjamin Paul Chamberlain, Josh Levy-Kramer, Clive Humby, and Marc Peter Deisenroth. 2018. Real-time community detection in full social networks on a laptop. *PloS one* 13, 1 (2018), e0188702.

[6] Wei-Cheng Chang, Yuexin Wu, Hanxiao Liu, and Yiming Yang. 2017. Cross-Domain Kernel Induction for Transfer Learning.. In *AAAI*. 1763–1769.

[7] Swarup Chattopadhyay, Tanmay Basu, Asit K Das, Kuntal Ghosh, and Late CA Murthy. 2020. Towards effective discovery of natural communities in complex networks and implications in e-commerce. *Electronic Commerce Research* (2020), 1–38.

[8] Laxmi Chaudhary and Buddha Singh. 2020. Community detection using maximizing modularity and similarity measures in social networks. In *Smart Systems and IoT: Innovations in Computing*. Springer, 197–206.

[9] Wei Cheng, Xiang Zhang, Zhishan Guo, Yubao Wu, Patrick F Sullivan, and Wei Wang. 2013. Flexible and robust co-regularized multi-domain graph clustering. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 320–328.

[10] Peter Chin, Anup Rao, and Van Vu. 2015. Stochastic block model and community detection in sparse graphs: A spectral algorithm with optimal rate of recovery. In *Conference on Learning Theory*. 391–423.

[11] Scott Emmons and Peter J Mucha. 2019. Map equation with metadata: Varying the role of attributes in community detection. *Physical Review E* 100, 2 (2019), 022301.

[12] Aleksandr Farseev, Ivan Samborskii, Andrey Filchenkov, and Tat-Seng Chua. 2017. Cross-domain recommendation via clustering on multi-layer graphs. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 195–204.

[13] Sharon Fogel, Hadar Averbuch-Elor, Jacov Goldberger, and Daniel Cohen-Or. 2018. Clustering-driven Deep Embedding with Pairwise Constraints. *arXiv preprint arXiv:1803.08457* (2018).

[14] Santo Fortunato and Darko Hric. 2016. Community detection in networks: A user guide. *Physics Reports* 659 (2016), 1–44.

[15] Zheng Gao, Gang Fu, Chunping Ouyang, Satoshi Tsutsui, Xiaozhong Liu, Jeremy Yang, Christopher Gessner, Brian Foote, David Wild, Ying Ding, and others. 2019. edge2vec: Representation learning using edge semantics for biomedical knowledge discovery. *BMC bioinformatics* 20, 1 (2019), 306.

[16] Zheng Gao, Chun Guo, and Xiaozhong Liu. 2019. Efficient personalized community detection via genetic evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 383–391.

[17] Zheng Gao and Xiaozhong Liu. 2017. Personalized community detection in scholarly network. *iConference 2017 Proceedings Vol. 2* (2017).

[18] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[19] Olivier Guédon and Roman Vershynin. 2016. Community detection in sparse networks via Grothendieck's inequality. *Probability Theory and Related Fields* 165, 3-4 (2016), 1025–1049.

[20] Nandinee Fariah Haq, Mehdi Moradi, and Z Jane Wang. 2019. Community structure detection from networks with weighted modularity. *Pattern Recognition Letters* 122 (2019), 14–22.

[21] Yen-Chang Hsu, Zhaoyang Lv, and Zsolt Kira. 2017. Learning to cluster in order to Transfer across domains and tasks. *arXiv preprint arXiv:1711.10125* (2017).

[22] Mohamed-Hamza Ibrahim, Rokia Missaoui, and Abir Messaoudi. 2019. Detecting Local Community Structures in Social Networks Using Concept Interestingness. *arXiv preprint arXiv:1902.03109* (2019).

[23] Zhuoren Jiang, Zheng Gao, Jinjiong Lan, Hongxia Yang, Yao Lu, and Xiaozhong Liu. 2020. Task-Oriented Genetic Activation for Large-Scale Complex Heterogeneous Graph Embedding. In *Proceedings of The Web Conference 2020*. 1581–1591.

[24] Masoumeh Kheirkhahzadeh, Andrea Lancichinetti, and Martin Rosvall. 2016. Efficient community detection of network flows for varying Markov times and bipartite networks. *Physical Review E* 93, 3 (2016), 032309.

[25] Florent Krzakala, Cristopher Moore, Elchanan Mossel, Joe Neeman, Allan Sly, Lenka Zdeborová, and Pan Zhang. 2013. Spectral redemption in clustering sparse networks. *Proceedings of the National Academy of Sciences* 110, 52 (2013), 20935–20940.

[26] Jiyi Li, Yukino Baba, and Hisashi Kashima. 2018. Simultaneous Clustering and Ranking from Pairwise Comparisons.. In *IJCAI*. 1554–1560.

[27] Xiaozhong Liu, Xing Yu, Zheng Gao, Tian Xia, and Johan Bollen. 2016. Comparing community-based information adoption and diffusion across different microblogging sites. In *Proceedings of the 27th ACM Conference on Hypertext and Social Media*. 103–112.

[28] Zhiwei Liu, Lei Zheng, Jiawei Zhang, Jiayu Han, and Philip S Yu. 2019. JSCN: Joint Spectral Convolutional Network for Cross Domain Recommendation. *arXiv preprint arXiv:1910.08219* (2019).

[29] Mingsheng Long, Jianmin Wang, Guiguang Ding, Dou Shen, and Qiang Yang. 2014. Transfer Learning with Graph Co-Regularization. *IEEE Trans. Knowl. Data Eng.* 26, 7 (2014), 1805–1818.

[30] Dongsheng Luo, Jingchao Ni, Suhang Wang, Yuchen Bian, Xiong Yu, and Xiang Zhang. 2020. Deep Multi-Graph Clustering via Attentive Cross-Graph Association. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 393–401.

[31] Andrea Montanari and Subhabrata Sen. 2016. Semidefinite programs on sparse random graphs and their application to community detection. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. 814–827.

[32] Evangelos E Papalexakis, Leman Akoglu, and Dino Ience. 2013. Do more views of a graph help? community detection and clustering in multi-graphs. In *Proceedings of the 16th International Conference on Information Fusion*. IEEE, 899–905.

[33] Ankita Shukla, Gullal Singh Cheema, and Saket Anand. 2018. ClusterNet: Semi-Supervised Clustering using Neural Networks. *arXiv preprint arXiv:1806.01547* (2018).

[34] Fan-Yun Sun, Meng Qu, Jordan Hoffmann, Chin-Wei Huang, and Jian Tang. 2019. vGraph: A Generative Model for Joint Community Detection and Node Representation Learning. *arXiv preprint arXiv:1906.07159* (2019).

[35] Meng Tang, Dmitrii Marin, Ismail Ben Ayed, and Yuri Boykov. 2019. Kernel cuts: Kernel and spectral clustering meet regularization. *International Journal of Computer Vision* 127, 5 (2019), 477–511.

[36] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community preserving network embedding. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[37] Xinghua Wang, Zhaohui Peng, Senzhang Wang, S Yu Philip, Wenjing Fu, and Xiaoguang Hong. 2018. Cross-domain recommendation for cold-start users via neighborhood based feature mapping. In *International Conference on Database Systems for Advanced Applications*. Springer, 158–165.

[38] Jaewon Yang and Jure Leskovec. 2013. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 587–596.

[39] Liang Yang, Xiaochun Cao, Dongxiao He, Chuan Wang, Xiao Wang, and Weixiong Zhang. 2016. Modularity Based Community Detection with Deep Learning.. In *IJCAI*. 2252–2258.

[40] Yen-Yun Yu, Shireen Y Elhabian, and Ross T Whitaker. 2018. Clustering With Pairwise Relationships: A Generative Approach. *arXiv preprint arXiv:1805.02285* (2018).