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

Zheng Gao  
Indiana University Bloomington  
gao27@indiana.edu

Zhuoren Jiang  
Zhejiang University  
jiangzhuoren@zju.edu.cn

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

Xiaozhong Liu  
Indiana University Bloomington  
liu237@indiana.edu

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 usersâ€™ 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 â€” 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 information to optimize the community structure of 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

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
Inspired by aforementioned discussion, we propose a novel problem - 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).

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

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.

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 detect 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 $u_i$ should have closer, similar or farther community closeness to user $u_j$. 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.
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:

$$L(\pi) = \sum_{i=1}^{m} q_i^\pi H(Q) + \sum_{i=1}^{m} p_i^\pi H(P^i)$$  \hspace{1cm} (1)$$

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

In the end, given a user $u_i$, we obtain its one-hot community representation $v_i^M$ 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:

\[
    v_{e,i}^G = \tanh(W_e^G p_i^e + b_e^G)
\]  

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 \( \pi \) 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 \( H^M = \{h_1^M, \ldots, h_n^M\} \) 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 manner:

\[
    a_j^G = (u_p^G)^\top \tanh(W_p^G h_j^G + b_p^G)
\]

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:

\[
    v_{p,i}^M = \sum_{n_j \in N^M(u_i)} \text{softmax}(a_j^M w_{\pi(n_j)}^G) h_j^M
\]

\[
    v_{p,i}^S = \sum_{n_j \in N^S(u_i)} \text{softmax}(a_j^S h_j^S)
\]

Both \( v_{p,i}^M \) and \( v_{p,i}^S \) are the weighted sum of the neighbour object representations. \( N^S(n_i) \) and \( N^M(n_i) \) denote user connected objects 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 \( c_{i,j}^G \) 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:

\[
    c_{i,j}^G = \sigma(u_c^G)^\top (\tanh(v_{c,i}^G) \ast d_j^G)
\]

The dot product between transformed user propagative representation \( \tanh(v_{c,i}^G) \) 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:

$$s^G = \sigma(W^G_0 [v^G_{i1}, d^G_{i1}] + b^G_0)$$
$$z^G = \text{tanh}(W^G_z v^G_{i1} + b^G_z)$$
$$d^G_{ij} = (1 - s^G) \ast d^G_i + s^G \ast z^G$$

where $G \in \{M, S\}$, $s^G$ denotes the update rate and $z^G$ is transformed user information to be updated in current community. $d^G_{ij}$ denotes the updated representation of the $j_{th}$ community in graph $G$, which is the weighted sum on previous community and current user representations. $W^G_{0}$ and $W^G_{z}$ denote related weight matrices, while $b^G_0$ and $b^G_z$ 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:

$$\mathcal{L}_c^G = \frac{1}{2K^2} \sum_{i,j} \cos(d^G_{i1}, d^G_{j1})$$

where $K$ is the number of communities in each graph.

2.5 Pairwise Community Detection

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

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

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 $r^G_{ij}$ between $u_j$ and $u_k$ in single graph $G \in \{M, S\}$:

$$r^G_{ij} = W^G_r (v^G_{i1} - v^G_{j1}) + b^G_r$$

where $W^G_r$ and $b^G_r$ are related weight and bias, respectively.

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

$$r_{ij} = \text{tanh}([r^S_{ij}, r^M_{ij}])$$

Similarly, the cross-graph correlation representation between $u_j$ and $u_k$ is calculated as $r_{jk}$.

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

$$\hat{y}_{i,j,k} = \sigma(W_0 (r_{ij} - r_{jk}) + b_0)$$

where $W_0$ and $b_0$ are the related weight and bias. $\sigma(\cdot)$ denotes the sigmoid activation function. In the end, the final loss $\mathcal{L}_{\text{total}}$ 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:

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

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 $\rho$ 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.

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

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 $(u_i, u_j, u_k)$ 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 Matthews Correlation Coefficient (MCC) 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.

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

---

### Table 1: Statistics of Two cross-graph datasets.

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

1. https://github.com/mapequation/infomap
2. http://yangliang.github.io/code/
3. https://github.com/melifluos/LSH-community-detection
4. https://github.com/eduarc/WMW
5. https://github.com/snap-stanford/snap/
which component has the largest impact. There are six components:

- DTR (Direct Transformation Representation)
- NF (Node-level filter)
- CF (Community Constraint)
- MT (Masked Training)
- RCR (Community Representation in the main graph)
- CC (Community-level Constraint)

### Table 2: All model performances for three different graph combinations in two datasets

| 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  | 0.3021 | 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   |

Among the six components, if we remove node-level filter, the performance drop dramatically, even worse than some of baseline models. 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 significantly beats the best baseline with $p < 0.01$.

### 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 (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 firmly to demonstrate their performance difference compared to the original model. All comparison results are showed in Table 3.

| 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. "-\( \ast \)" refers to remove related component from our model. Results are scaled in percentage ($\%$).
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 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.

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 δ = {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 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 illustrated in the right part of Figure 6.

From the right-part table, $u_1$ and $u_2$ share similar shopping interests in clothing products. $u_3$ is more like a fashion girl. While the shopping interests of $u_3$ are 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 $c_7$ 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 compared 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, Ayon Chen, Peter J Bickel, Elizaveta Levina, et al. 2013. Pseudo-likelihood methods for community detection in large sparse networks. The Annals of Statistics 41, 4 (2013), 2097–2122.

[2] Aditya Balád, Willon Kion-Crosby, and Alexandre Morozev. 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 Leon, and Jonatan Gonzalez. 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. Plosone 13, 1 (2018), e1088702.

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

[7] Swapru Chattopadhyay, Tammay Basu, Arit 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 Buddhika 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, Zhuhan Guo, Yuhua Wu, Patrick F Sullivan, and Yiming Yang. 2017. Stochastic block model and community detection in sparse graphs: A spectral algorithm with optimal rate of recovery. In Conference on Learning Theory. 391–423.

[10] Scott Emmons and Peter J Mucha. 2015. Map equation with metadata: Varying communities to generate pseudo labels for training in a separate procedure. In Proceedings of the Genetic and Evolutionary Computation Conference. 383–391.

Zheng Gao and Xiaohong Liu. 2017. Personalized community detection in scholarly network. InConference 2017 Proceedings Vol. 2 (2017).

[11] 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.

[12] 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.

[13] Peter Chin, Anup Rao, and Van Vu. 2015. Structured component analysis: A nonnegative matrix factorization approach. In Proceedings of the 27th ACM Conference on Hypertext and Social Media. 103–112.

[14] 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.

[15] 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 19th International Conference on Web Search and Data Mining. 393–401.

[16] Anirban Chakraborty, Vinita Kumbhakar, and Saikat Ray. 2020. Semi-Supervised Clustering Using Neural Networks. arXiv preprint arXiv:1806.01547 (2018).

[17] Tianlin Shu and Jure Leskovec. 2020. Transfer Learning with Graph Co-regularization. IEEE Trans. Knowl. Data Eng. 26, 7 (2014), 1805–1818.

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

[19] Zheng Gao, Gang Fu, Chunping Ouyang, Satoshi Tsutsui, Xiaozhong Liu, Jeremy Emmons and Peter J Mucha. 2019. Map equation with metadata: Varying communities to generate pseudo labels for training in a separate procedure. In Proceedings of the Genetic and Evolutionary Computation Conference. 383–391.