COMMUNITY DETECTION IN GRAPHS

Zheng Gao

Submitted to the faculty of the University Graduate School

in partial fulfillment of the requirements

for the degree

Doctor of Philosophy

in the Luddy School of Informatics, Computing, and Engineering,

Indiana University

June 2020

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Doctoral Committee

Xiaozhong Liu, Ph.D. (Principle Advisor)

Xiaofeng Wang, Ph.D.

Yong-Yeol Ahn, Ph.D.

Kahyun Choi, Ph.D.

Date of Defense: 05/22/2020

ACKNOWLEDGMENTS

I have been in the United States for seven years and in the quiet and beautiful down of Bloomington, Indiana for five years to accomplish this doctoral program. I have never imagine that time can go that fast. Being a scientist was my dream when I was a child, it is so excited for me to be so close to reach it now. Doctoral life is long and mixed with depression and pressure, while it also associated with the happiness when my sweat irrigates published papers. I am really glad that I can achieve something before my thirty-year old birthday.

Thank my advisor, Xiaozhong Liu, who patiently brings me from nobody to a guy with certified academic and industrial experiences. I will always follow his edification throughout my future career and try my best to be outstanding. Also big thanks to all my committee members to offer me supportive suggestions about how to criticize researches, draft papers and present works.

Thank my parents and girlfriend who support me since the very beginning of my study aboard. I barely have time and opportunity to accompany with them due to the long long distance from the United States to China. Their comprehension is my solid back-support to overcome all challenges occurred in these years.

Thank all colleagues I met during my two internships at Alibaba and Amazon. I was inspired so much from these talented people in terms of how to do academic researches and be professional in industry. Hanging out with them upgrades myself significantly.

Finally, I hope the pandemic can be gone soon and people can restart their works. I am so grateful for what I have and can't bear to lose anything more from the pandemic. Truly wish a peaceful world without suffering any pain.

Zheng Gao

COMMUNITY DETECTION IN GRAPHS

Community detection has always been one of the fundamental research topics in graph mining. As a type of unsupervised or semi-supervised approach, community detection aims to explore node high-order closeness by leveraging graph topological structure. By grouping similar nodes or edges into the same community while separating dissimilar ones apart into different communities, graph structure can be revealed in a coarser resolution. It can be beneficial for numerous applications such as user shopping recommendation and advertisement in e-commerce, protein-protein interaction prediction in the bioinformatics, and literature recommendation or scholar collaboration in citation analysis.

However, identifying communities is an ill-defined problem. Due to the No Free Lunch theorem [1], there is neither gold standard to represent perfect community partition nor universal methods that are able to detect satisfied communities for all tasks under various types of graphs. To have a global view of this research topic, I summarize state-of-art community detection methods by categorizing them based on graph types, research tasks and methodology frameworks. As academic exploration on community detection grows rapidly in recent years, I hereby particularly focus on the state-of-art works published in the latest decade, which may leave out some classic models published decades ago.

Meanwhile, three subtle community detection tasks are proposed and assessed in this dissertation as well. First, apart from general models which consider only graph structures, personalized community detection considers user need as auxiliary information to guide community detection. In the end, there will be fine-grained communities for nodes better matching user needs while coarser-resolution communities for the rest of less relevant nodes. Second, graphs always suffer from the sparse connectivity issue. Leveraging conventional models directly on such graphs may hugely distort the quality of generate communities. To tackle such a problem, cross-graph techniques are involved to propagate external graph information as a support for target graph community detection. Third, graph community structure supports a natural language processing (NLP) task to depict node intrinsic characteristics by generating node summarizations via a text generative model.

The contribution of this dissertation is threefold. First, a decent amount of researches are reviewed and summarized under a well-defined taxonomy. Existing works about methods, evaluation and applications are all addressed in the literature review. Second, three novel community detection tasks are demonstrated and associated models are proposed and evaluated by comparing with state-of-art baselines under various datasets. Third, the limitations of current works are pointed out and future research tracks with potentials are discussed as well.

Xiaozhong Liu, Ph.D. (Principle Advisor)

Xiaofeng Wang, Ph.D.

Yong-Yeol Ahn, Ph.D.

Kahyun Choi, Ph.D.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Definition of Community Detection

In the beginning, it is necessary to introduce the terminology used in this dissertation. A graph, in its intuitive understanding, is a group of nodes connected with each other to form either one connected or several disjoint components. In this thesis, a graph is also called a network, and a node or vertex (plural: vertices) is a single object appearing in the graph. The terms edge or link refers to the connections between two nodes. Graph mining, also known as complex network analysis, is the practice of exploring graphs to analyze their properties and, often, to extract them automatically. These terms are often used interchangeably, and there is no fundamental difference between them. Community detection is a subtopic within graph mining that aims to detect node closeness relationships given the graph topological structure. Also known as clustering, it entails grouping nodes into communities (also called groups or clusters) where the within-community nodes should have as many connections as possible and between-community nodes should have as few connections as possible.

Mathematically, given a graph $G(V, E)$ where V is the node set and E is the edge set, community detection approaches aim to find a partition $C = \{c_1, c_2, ..., c_k\}$ to group all nodes into k communities, where c_i refers to the i_{th} community with N_i nodes. For the simplest scenario in which a node can only belong to one community ($v_i \in c_j$, $N = \sum_{i=1}^{k} N_i$) is the number of total nodes $|V|$ in the graph.

A simple visualization of graph community structures is shown in Figure 1.1 where nodes and edges constitute the graph and different colors refer to different communities. It is easy to see that there are three communities, which are marked in different colors (blue, green, red). Within each community, the nodes are densely connected with each other, whereas there are few edges between

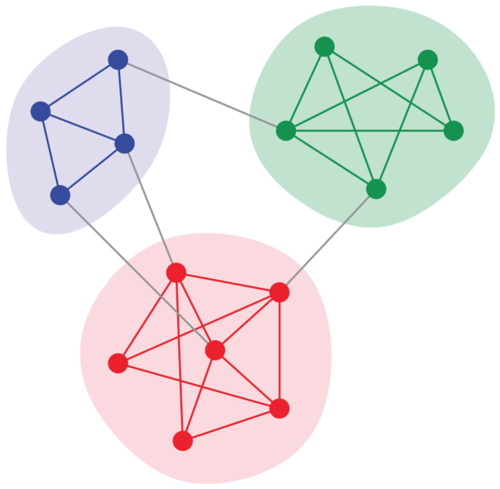


Figure 1.1: Example graph showing community structure. The nodes of this graph are divided into three communities, with most edges falling within communities and only a few between communities. The figure is contributed from [2].

nodes in different communities.

The goal of community detection is to assign each node to a community (disjoint community detection) or multiple communities (overlapping community detection) in a manner that satisfies the abovementioned criteria. Apart from the simple concept definitions introduced above, a Table 1.1 provides a comprehensive glossary of terminology used in this thesis, introducing representative symbols and brief definitions.

1.2 Background and Motivation

Graph mining is one of the fundamental research areas in artificial intelligence; in the recent decade, community detection continues to be an essential topic within graph mining. Community detection approaches group nodes into communities, which can be regarded as a coarser view of a graph and reveal the latent knowledge of nodes. This detected knowledge can be used to support further tasks such as link prediction or node profile construction. For example, detecting researcher communities in a scholarly graph can imply future collaboration between them, or offer paper/research recommendations to targeted users. In biology, community structure can be used to infer possible protein-protein binding relationships, which may potentially contribute to health-

Term	**Definition**
$G(V, E)$	Graph G with node set V and edge set E.
Ω	$\Omega = \{\omega_1, \omega_2, ..., \omega_k\}$ is the ground truth community partition by models where ω_k is the k_{th} predicted community.
C	$C = \{c_1, c_2, ..., c_j\}$ is the detected community partition where c_j is the j_{th} community.
N, M	$N = \|V\|$ denotes the node number and $M = \|E\|$ denotes the edge number in graph G
N_c	$N_c = \|c\|$ is the number of nodes in community c
$N_{c,\omega}$	$N_{c,\omega} = \|c \cap \omega\|$ is the common nodes in the detected community c and ground truth community ω
E_c^{in}	The number of edges within the community c. $E_c^{in} = \{(u, v) \in E : u \in c, v \in c\}$
E_c^{out}	The number of edges on the boundary of the community c. $E_c^{out} = \{(u, v) \in E : u \in c, v \notin c \textit{ or } u \notin c, v \in c\}$
$D(v)$	Degree of node $v \in V$
D_c	The sum of node degrees in community c
$\mathcal{N}(v)$	The neighbour node set of node v
A	Graph adjacent matrix

Table 1.1: Glossary of technical terms.

care. Therefore, exploring community detection tasks has huge practical value in both academic and industrial scenarios.

The popularity of community detection research is suggested by the thousands of papers published on the subject in top-tier journals and presented at conferences. Among these, some are survey papers which summarize state-of-the-art models from different perspectives; these have served as great references for me to organize relevant papers into different categories. [3, 4, 5] are general survey papers which broadly introduce the definition of community detection and several types of models. Other papers pay more attention to detecting communities in particular types of graph. For example, [6] is a classic paper which summarizes community detection in directed graphs; [7] particularly focuses on large-scale graphs with empirical studies; and [8] particularly summarizes the latest community detection models for multi-layer graphs. Still other papers focus on particular types of model frameworks. For example, [9] is the most recent work on block models in community detection. [10, 11] are both authentic survey papers on overlapping community detection, even though they are slightly dated as of 2020. [12] is a classic research work that introduces a collection of spectral clustering models. To demonstrate how community detection supports interdisciplinary research efforts, [13] and [14] show the application of community models to social media mining as well as other domains such as biology and neuroscience. Finally, [15] summarizes all types of evaluation metrics for model performance comparison.

However, all of the aforementioned papers are either slightly out of date or relevant only to a specific topic; what is lacking is an effort to summarize and track the latest works from all possible main perspectives. This inspired me to include the most representative papers in this thesis, categorized by my own defined taxonomy. Specifically, derived from all existing works, I aim to leverage in-depth analysis in some subtle tasks and propose novel models to improve current model performance. A survey of the existing literature also inspired me to formulate and solve the various research questions mentioned in the following section.

1.3 Research Problems

In this section,I introduce three motivating tasks which this thesis aims to solve and briefly discuss my proposed approaches. The first task is *Personalized Community Detection*, in which communities with different resolutions are formed given user information need. The second task is *Cross-Graph Community Detection*, in which I detect pairwise user community closeness in sparse graphs by leveraging propagated information from external graphs. The third task is *Community-Aware Dynamic Product Summarization*, in which community is applied as supportive information to generate temporal product summarizations across different time periods. Details of my proposed models, along with extensive experiments and discussions, are introduced in the following chapters.

1.3.1 Personalized Community Detection

From a user-centric viewpoint, the ideal communities should provide a high-resolution partition in areas of the graph relevant to the user's needs and a coarse partition on the remaining areas so as to best respond to the user's needs (which can be formulated as a "query") in concentrated areas

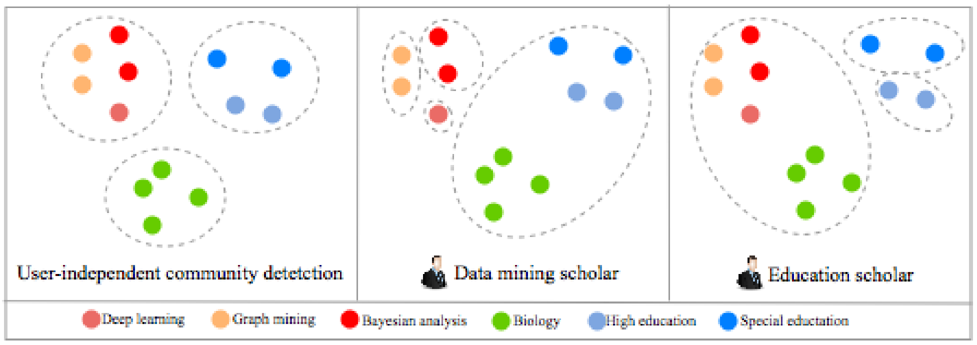


Figure 1.2: An example of personalized community detection on a scholarly graph

while leaving irrelevant areas less precisely specified.

For instance, in Figure 1.2, two different scholars in *education* and *data mining* domains, respectively, may consume the same scholarly graph differentlybecause they may need more detailed community exploration in their own domains whereas generalized community information is sufficient in irrelevant domains. For instance, a *data mining* scholar may need more detailed communities in areas such as deep learning, graph mining, and Bayesian analysis, but for an *education* researcher it may suffice to generalize these as computer science or even just science.

To solve this problem, I propose a **g**enetic **P**ersonalized **C**ommunity **D**etection (gPCD) model with an offline and an online step to efficiently cope with the time complexity issue in personalized community detection. Details of the model structure are given in Chapter 3.

1.3.2 Cross-Graph Community Detection

For a vulnerable graph with sparse connectivity, however, existing community detection algorithms can hardly probe enough information to optimize the community structure. Unfortunately, in real cyberspace this is a very common problem: while a handful of giant players (e.g., Google, Facebook, and Amazon) maintain high-quality graphs, thousands of apps are suffering from a "cold-start" problem in this area. If many users are isolated because of data sparseness, I can hardly determine any community information.

To cope with this challenge, in this thesis I propose a novel research problem: Cross-Graph

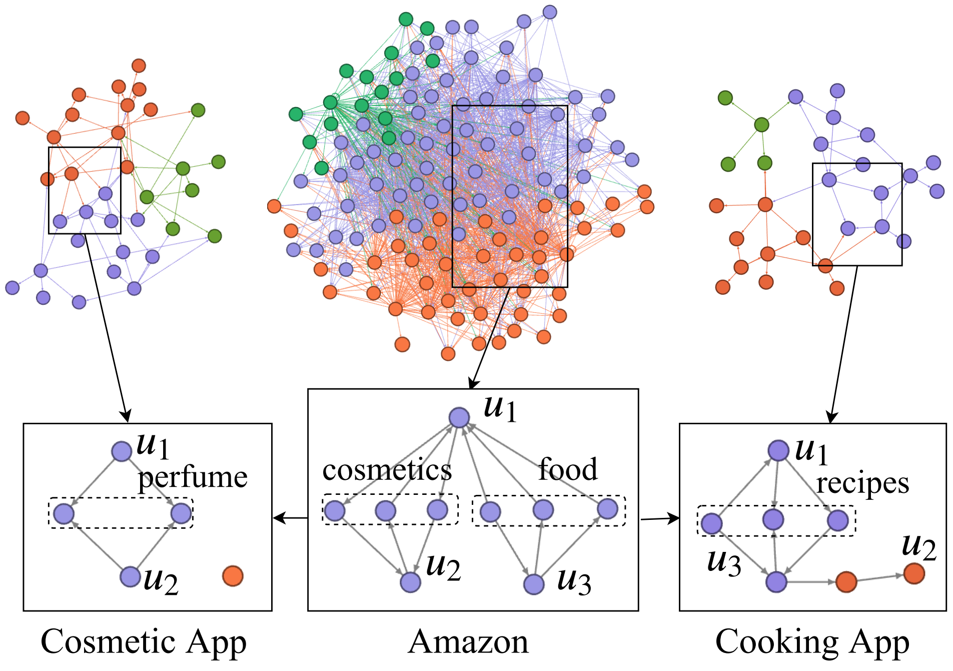


Figure 1.3: The Amazon graph is a main shopping graph while the Cosmetic App and the Cooking App are two small sparse graphs. Those graphs share mutual users in which three of them are selected to demonstrate their behaviors. Node colors indicate their related communities.

Community Detection. The idea is based on the fact that an increasing number of small apps are utilizing user identity information inherited from giant providers; for instance, users can easily log in to a large number of new apps using Facebook and Google ID. In such an ecosystem, the large main graph can provide critical information to enlighten community detection in many small sparse graphs. Figure 1.3depicts an example in which the Cooking and Cosmetic apps inherit important topological information from the main Amazon graph for enhanced community detection.

Inspired by the foregoing discussion, I 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 $\langle u_i, u_j, u_k \rangle$, I aim to predict the pairwise community relationships, e.g., compared with user u_k, does user u_j have closer, similar or farther community closeness to user u_i? The details of this model structure are mentioned in Chapter 4.

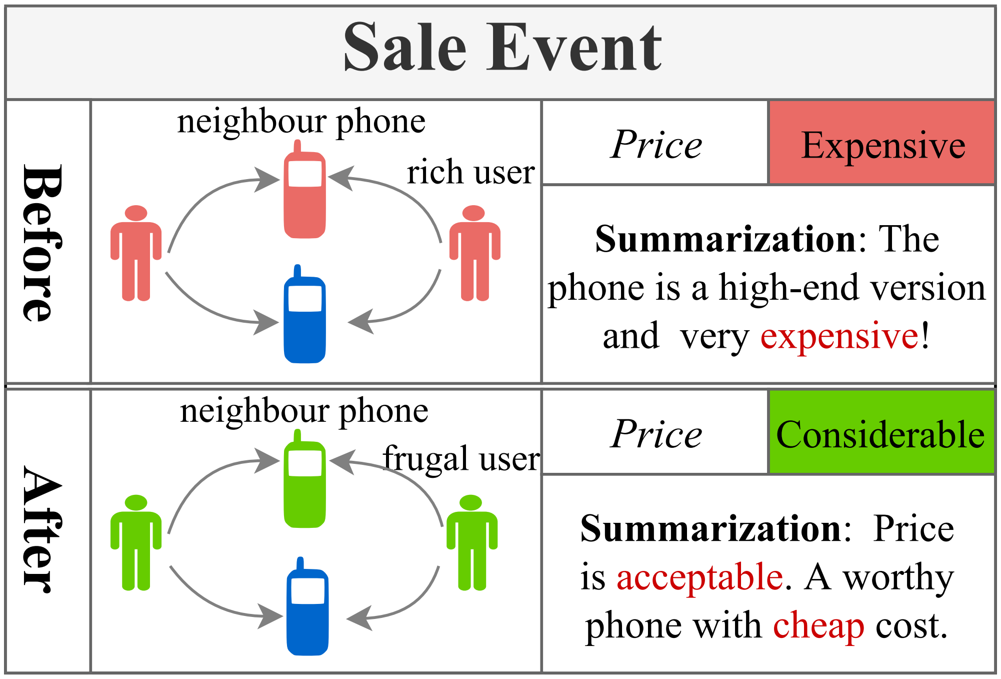

Sale Event
Before
neighbour phone
rich user
Price
Expensive
Summarization: The phone is a high-end version and very expensive!
After
neighbour phone
frugal user
Price
Considerable
Summarization: Price is acceptable. A worthy phone with cheap cost.

Figure 1.4: An example to illustrate how to dynamically detect communities and select within-community neighbor products (red phone → green phone) for depicting current product (blue phone) characteristic change before & after a sale event from user behaviors.

1.3.3 Community-Aware Dynamic Product Summarization

Product communities can implicitly reflect product characteristics. Therefore, understanding product communities and tracking their changes in a timely manner can support better decision-making for commercial purposes, such as informing online retailers in creating timely sales plans. Therefore, the *Dynamic Community-Aware Product Summarization* task is of great importance. Such summarization not only indicates products' dynamic membership changes but also utilizes these changes to depict product characteristics in readable contexts for easier interpretation.

However, existing generative models relying on reviews inevitably face the sparsity issue: few products can gain enough reviews in a short time to depict the product's temporal characteristics. User behaviors, on the other hand, are abundant and highly related to the product's nature. Therefore, instead of review-based product summarization, I aim to generate community-aware product summarization in a dynamic manner. These approaches share the same goal but have huge differences in terms of proposed models. Review-based summarization directly runs a sequence-to-sequence model to generate product summarization by simply leveraging review context information. A community-aware product summarization, however, is generated from within-community neighbor product reviews. This means that it is a two-step approach, requiring both community detection and summarization generation.

As Figure 1.4 depicts, when a sale event on a high-end phone brings frugal users' instant *clicks*, behavior-based algorithms can immediately consume this information and locate updated within-community neighbor products (red phone $\rightarrow$ green phone) whose sufficient reviews help to update the product (blue phone) summarization. For review-based approaches, accumulating enough reviews to characterize this dynamic change may take a longer time.

In the end, I propose a **B**ehavior based **D**ynamic **S**ummarization (BDS) model to accommodate user behavior for dynamic product summarization. The proposed model uses learned communities as a criteria to select neighbor products via a reinforcement approach, where textual review information is used to generate target product summarizations in a timely manner. Details of the model structure are given in Chapter 5.

1.4 Contribution

In this dissertation, I summarize a rich selection of existing works and propose three different community detection tasks. In the literature review, from the more than one thousand candidate papers published in the recent decade, I select around three hundred highly representative works and define my own taxonomy to categorize them. Compared with other survey papers (which are also briefly discussed in this dissertation), my literature review is better categorized and contains more comprehensive and state-of-the-art studies. It also highlights the application and evaluation of community detection, two topics which have largely escaped mention in previous surveys.

In terms of the first research task (personalized community detection), the contribution of this study is threefold:

- I address a novel personalized community problem and propose a model to generate communities of varying resolution in response to user needs.

- The proposed model contains an offline and an online step. The offline step takes charge of most calculation to enable an efficient online step: the construction of the binary community tree has a time complexity of $O(|V|^2)$ in the worst case where $|V|$ denotes the number of vertices in the graph; representation learning on the both binary community tree and user needs has the same time complexity as Node2vec [16]. The online genetic pruning step running under the parallel environment achieves $O(\frac{2^d KP}{M})$ time complexity where d denotes the depth of the tree, K denotes the community number, P denotes the initialized population size in the genetic approach and M denotes the number of Mappers/ Reducers in the Hadoop Distributed File System (HDFS).

- I evaluate the proposed model on a scholarly graph and a music graph. In my model, the offline step is separately calculated and remains unchanged once constructed, while the online step guides the personalized community detection. Hence I only compare the online step results with baselines' performance. Extensive experiments show that my model outperforms the baselines in terms of both accuracy and efficiency.

This paper's treatment of cross-graph community detection (the second task) makes four main contributions:

- I 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.

- My method departs from conventional community detection methods in exploring community structure from a pairwise viewpoint. In this way, I can efficiently deal with heterogeneous graph information and solve cold-start problem for users with no recorded 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 my model's superiority. I also evaluate my model's robustness on graphs with varied sparsity scales and conduct many other supplementary studies.

For the third problem, i.e., community-aware product summarization, the contribution of this work is threefold:

- To the best of my knowledge, this is the first effort to leverage user behavior for dynamic product summarizations. This work pioneers the investigation of behavior-based summarization.
- In my model, a reinforcement learning approach learns the sampling strategy on seed products with rewards from both community distribution (calculated from behavior-to-community prediction) and semantic (calculated from summarization generation) viewpoints. When generating product summarization, my model does not require the target product's reviews as an input; thus, it is able to solve the problem of review sparseness, even in a zero-review scenario.
- Experiments on a large e-commerce dataset show that my proposed model significantly outperforms the baselines from both automatic and human perspectives. Extensive studies also prove the efficacy of each model input component.

In the final chapter, I summarize the contributions of all previous chapters. Meanwhile, the limitations of this study and the community detection domain are discussed as well. To tackle these limitations, future studies are suggested, highlighting broader potential tracks and research problems in community detection. In particular, deep learning methods and disciplinary analysis are two classes of approach with the greatest potential.

1.5 Thesis Structure

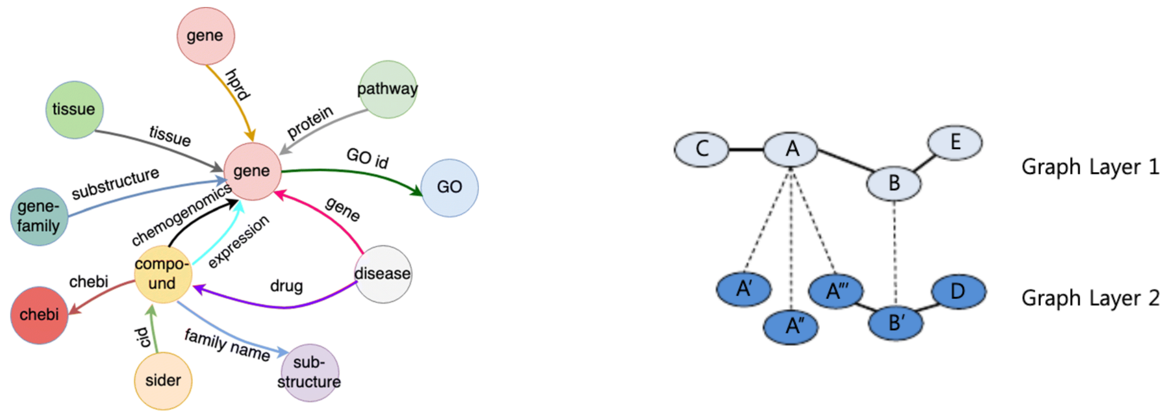
The remaining chapters of the thesis are organized as follows. In Chapter 2, it iss a literature review of the most representative papers published in the recent decade. Selected out of over one thousand papers, the studies mentioned cover the most popular topics in community detection following a well-defined taxonomy. I summarize papers from five different perspectives, including graph types, different tasks, main track methods, applications and evaluation. Within each section, I further organize relevant papers into different sub-categories. For example, the graph type section introduces papers dealing with community detection problems on heterogeneous & multi-layer graphs, sparse graphs, dynamic graphs, large graphs, and attribute graphs. In this hierarchical taxonomy, papers are best distinguished and separated into different topics.The Chapter 3 introduces the first proposed task: personalized community detection. I introduce the background and motivation of the research question, propose my own model, run experiments to compare with other state-of-the-art baselines, and discuss experimental results to infer some in-depth understandings. Chapter 4 and Chapter 5 follow a similar structure to introduce the other two proposed tasks, namely, cross-graph community detection and community-aware dynamic product summarization. Finally, in Chapter 6, I summarize the overall contribution of this dissertation in detail, point out current limitations in both this dissertation and the community detection domain as a whole, and suggest some ways of overcoming these limitations through future research.

CHAPTER 2
RELATED WORKS

In this chapter, I select the most outstanding studies based on a self-defined criteria (either published in a set of pre-selected venues or performed the highest impact by receiving at least fifty citations). To better introduce these papers in a well-organized manner, I categorize them into five following tracks borrowing ideas from aforementioned survey studies [3, 4, 5]. In detail, papers in the Graph Type track focus on detecting communities in different types of graphs, such as heterogeneous or sparse graphs. In the Task track, the selected studies aim to solve particular tasks in community detection, such as deciding the correct number of communities. In the Methodological track, the introduced studies solve the general community detection problem via different types of model frameworks such as Modularity or spectral methods. In the Application track, selected studies discuss community detection applications and how to apply them to other disciplines. In the last Evaluation track, it lists papers to summarize the evaluation metrics widely used for model justification and comparison.

2.1 Graph Types

Graph types are various and complex. Different type of graphs have their particular characteristics, which requires to be handled by special techniques. With the rapid development of graph mining, more and more complex graphs are constructed from real scenarios and analyzed by researchers, which offers the possibility to summarize several main types and relevant works in this section. Particularly, the following paragraphs introduce five main types of graph which are most frequently appeared in research studies and tightly connected with real-world scenarios, including heterogeneous & multi-layer graph, sparse graph, dynamic graph, large graph and attribute graph.



(1) heterogeneous graph (2) multi-layer graph

Figure 2.1: Figure in the left is a typical heterogeneous medical graph in which nodes and edges refer to medical entities and their binding relationships from [17]. Figure in the right is a multi-layer graph with two graph layers from [8].

2.1.1 Heterogeneous & Multi-layer Graph

The research works about heterogeneous and multi-layer graph community detection are summarized together. The reason is that both types of graphs share very similar structure but still keep their own characteristics. For a heterogeneous graph, it should contains more than one type of node or edges. While for a multi-layer graph, it is composed from multiple single layer graphs. A single layer graph is with only one type of node/edge. By connecting nodes across single layer graphs with edges, it formulates a multi-layer graph in the end. Figure 2.1 clearly shows the definition difference between heterogeneous graph and multi-layer graph. In fact, multi-layer graph can be regarded as a particular type of heterogeneous graph.

Heterogeneous graph, also called heterogeneous information network, is a particularly complex graph which is ubiquitously appeared in current research works. By definition, a heterogeneous graph $G(V, E)$ has an extra node mapping function $\phi : V \rightarrow \mathcal{A}$ and an edge mapping function $\psi : E \rightarrow \mathcal{R}$ where each node $v \in V$ belongs to a particular node type $\phi(v) \in \mathcal{A}$ and each edge $e \in E$ belongs to a particular edge type $\psi(e) \in \mathcal{R}$. $\mathcal{A}$ is the collection of all node types and $\mathcal{R}$ is the collection of all edge types. A heterogeneous graph contains multiple node/edge types, which means at least either $|\mathcal{A}| > 1$ or $|\mathcal{R}| > 1$.

[18] defines a graph structure called metapath, which is a path that connects node types with different types of edges and can be regarded as a general structure to represent path semantics. For example, a metapath in a scholarly graph can be “Organization-(affiliated with)$\rightarrow$Author-(publish at)$\rightarrow$Venue” where it contains three node types (“Organization”, “Author” and “Venue”) and two types of edges (“affiliated with” and “publish at”). Particularly in this study, it integrates metapath selection with user guided clustering (a small set of nodes whose communities are known as prior information). In the end, it aims to learn the weights on selected metapaths to best satisfy the prior community knowledge via an EM approach.

[19] introduces a joint approach to learn node affiliation distributions on popular communities in heterogeneous graph and detect node outliers which do not belong to any of the popular community distribution patterns in a holistic manner. It decomposes the heterogeneous graph into several homogeneous ones. The community detection step is derived from nonnegative matrix factorization framework and the refined outlier detection is learned via an iterative two-stage approach. In the NMF based community detection process, all prior-known community membership matrices of single node-type graphs $\{T_1, ..., T_k\}$ can be used to calculate node community distribution under each node type:

$$\underset{W,H}{\operatorname{argmin}} \sum_{i=1}^{k} ||T_i - W_i H_i||^2 + \alpha \sum_{i \le j \le k}^{k} ||H_i - H_j||^2 \tag{2.1}$$

which should subject to $W_i \ge 0$ and $H_i \ge 0$ for $\forall i \le k$. W_i is the node vector representation matrix in the i_{th} node-type graph and H_i is the related community centroid matrix. The later part is the regularized term.

A node outlier score is calculated as the distance ($Dist(\cdot)$) between the current node community membership and its nearest community centroid:

$$S(i) = \operatorname*{argmin}_{j} Dist(T_{k_{(i,\cdot)}}, H_{k_{(j,\cdot)}}) \tag{2.2}$$

Top ρ nodes with largest outlier score will be regarded as outliers in current step. In an iterative learning process, the NMF method is optimized by reducing the detected outliers from the whole graph to support new node outlier detection. It stops when no outlier nodes are found out.

[20] solves a heterogeneous and incomplete graph community detection problem where only partial attribute information are given. The output of this paper includes two parts: node soft clustering distribution to indicate the probability of node affiliations to communities, and weights of edge types appeared in the graph. In model optimization, all edge types are assigned same weights in the beginning. The clustering process thereafter is applied to update the edge type weights according to the calculated average edge type consistency, which in return guides the clustering process. The iterative process reaches to the end once all updated parameters are stable.

Instead of node community detection, [21] proposes an EM model for edge community detection. It aims to re-generate the whole graph by finding a partition to maximize the edge generation probabilities. Given an edge e_{ij} with start node i and end node j, its expected generation score in community z is calculated as $A_{ij}^z = w_z\theta_{iz}\theta_{jz}$. w_z is the size of community z and θ_{iz} is the probability that community z contains the node i. Then the probability of a link e_{ij} belongs to community z is calculated by dividing the whole generation score in all communities, as:

$$R_{ij}^z = \frac{A_{ij}^z}{\sum_k A_{ij}^k} \tag{2.3}$$

The edge e_{ij} will be assigned to the community with largest generation probability. And all parameters can be learned through an EM approach.

[22] presents a social influence based framework to detect user communities in social graphs. In this paper, a social graph represents user collaborations (authors are connected by collaboration relationships) and an activity graph represents associated activity similarities (venues are connected

by their topic similarities). The edges between social graph nodes and activity graph nodes represents their influence flow, such as publishing histories of authors towards venues. The three parts (social graph, activity graph and edges between them) constructs the final influence graph. First, it introduces a new metric to measure node similarity jointly using self-influence (social graph) and co-influence (influence graph) information. Later on, a designed SI-Cluster model takes advantage of the metric to learn node communities in the social graphs only via an Kmeans-like process.

OCD-HSN model [23] detect node communities in four main steps. First, a set of seed nodes are selected according to their local minimum conductance, which enables a parallel community computation. Second, overlapping community detection is applied using local node expansion (adding and removing nodes iteratively). Third, node degrees are recalculated and normalized. Fourth, current communities are regarded as a node with coarser resolution to construct the next-level graph. Followed by all the previous steps, a hierarchical community partition can be formed in the end.

[24] proposes a semi-supervised model to detect attribute heterogeneous graph communities by considering three types of information: node attributes, graph topological structure, and prior knowledge of must-links and cannot-links. It first calculates and assigns weights to node pairwise attribute similarity matrix and connected similarity matrix (calculated from meta-paths) to construct a node similarity matrix S. Then it considers the constraints (must-links and cannot-links) by assigning penalties to related node pairs. In the end, the unified model is optimized through an Expectation-Maximization (EM) framework to learn all the model parameters.

Muilti-layer graph $G = \{G^1, G^2, ..., G^k\}$ is composed by k single-layer graphs $G^k = (V^k, E^k)$. For each possible pairwise single-layer graphs G^i and G^j, there is a mapping function $\phi : G^i \rightarrow G^j$, meaning the nodes in graph G^i can link to nodes in graph G^j, which formulates the connections between single-layer graphs. An extreme case of a multi-layer graph is that each single layer graph contains the same nodes.

[8] is a short survey paper to introduce the definition of mullti-layer graph and differentiate its concept from heterogeneous graph. It offers a bunch of multi-layer graph datasets and points out

several main tracks of methods, including pattern mining and matrix factorization. Typically, it introduces a popular two-layer graph community detection problem and introduces six main track methods, including cluster expansion, matrix factorization, unified distance, model-based methods, pattern mining and graph merging.

[25] constructs multi-layer graphs by tracking node dynamic changes across time. Therefore, each single layer graph shares the same nodes whose status and connections are changed over time. To solve the community detection in dynamic graphs, it generalizes the original Modularity method to adapt multi-layer graphs. It also defines a persistence metric which measures the graph changes and stability across time.

Multiplex Infomap [26] is derived from existing Infomap algorithm which originally designed for single-layer graphs. It uses codebooks to index each node and community in each layer, so that to record the paths of a random walker wandering in the multi-layer graph. The best community structure should obtains the shortest description length for the random paths. Similarly, [27] is also a generalized model from existing methods which is originally designed for single-layer graph community detection. Within the paper, it considers the simplest two-layer graph. It defines two type of models including an AND graph which only contains edges appeared in both layers, and a OR graph which contains edges appeared in either layers. In the end, it learns the two community partition result C_1 and C_2 in individual layers to maximize the probability to generate the AND graph from the two partitions and OR graph.

[28] develops a multi-layer Modularity model by utilizing a defined graph structure called Harmonic Motif. By definition, a harmonic motif is a dense graph with largest average Z-score values in all k graph layers. The Z-score shows the statistical significance of a subgraph towards the whole graph, which is calculated as:

$$Z = \frac{N_{real} - mean(N_{rand})}{std(N_{rand})} \tag{2.4}$$

where N_{real} is the number of time that the subgraph appears in each layer. $mean(N_{rand})$ and

$std(N_{rand})$ refers to the mean and standard derivation of the number of times that the subgraph appears in a random graph with same node and number of edges.

From k graph layers, the paper extracts all harmonic motifs to form a new graph layer. The existing k layers are in the end coupled with the new layer to detect communities in the new graph layer by maximizing the integrated Modularity.

2.1.2 Sparse Graph

Sparse graph is a very common phenomenon in graph mining, which is in contrary to dense graphs. However, their is no clear distinction between dense graph and sparse graph. A common notion is that dense graph is with the number of edges as $|E| = \mathcal{O}(|V|^2)$ as the maximum number of edges in a directed graph is $|V|(|V|-1)/2$. And a sparse graph usually contains the number of edges as $|E| = \mathcal{O}(|V|)$. Due to the fact that most of constructed graphs in real-world scenarios are sparse and incomplete, understanding sparse graph community detection gains a lot of practical value.

Due to the fact that spectral clustering methods are not able to perform well in sparse graphs, [29] argues that spectral methods on a refined matrix based on non-backtracking walks may achieve much better results than on the original adjacency matrix or Laplacian matrix. The non-backtracking matrix B is a $2|E| \times 2|E|$ matrix, defined as follows:

$$B_{(u \to v),(w \to x)} = \begin{cases} 1 & \text{if v = w and u} \neq \text{x} \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

[30] designs a model for undirected, unweighted sparse graph community detection. It points out two types of clustering errors: *(a)* an edge between nodes in different communities and *(b)* a missing edge between nodes in the same community. These errors can be represented as an error matrix S with corresponding values $\{+1, -1, 0\}$. To treat the two types of errors differently, a cost matrix M is utilized as well by assigning different weights on different error types. The object

function thereafter is as follows to minimize the two types of errors:

$$\underset{C,S}{\operatorname{argmin}} ||C|| + ||M \cdot S|| \tag{2.6}$$

with the restraints that $0 \leq C_{ij} \leq 1$. C is the cluster matrix where $C_{ij} = 1$ if v_i and v_j are in the same community, and 0 otherwise.

[31] claims involving perturbations in spectral clustering can improve its model performance in sparse graphs. The proposed model adds pseudo week edges between nodes to connect all disjoint components which suppose to be in the same community. Particularly, it adds $0.25 * (\lambda/|V|)$ where λ is the average node degree in the sparse graph to each datapoint of the adjacency matrix A.

[32] combines a perturbation strategy and link prediction in sparse graphs to assess all communities and select the most significant ones. As triangle is the smallest structural unit in a graph, the paper decides to complete partial of the open triangles in the original graph to achieve a denser graph. An open triangle completion means if there are two edges e_{ij} and e_{ik} links between a selected node triplet $< v_i, v_j, v_k >$, it will add an edge e_{jk} to the original graph. By using this enhancing method, better community partition can be achieved from the denser graph.

A rich number of researchers tend to explore sparse graph community detection from mathematical perspective. [33] theoretically proves the lower and upper bound of information-theoretic threshold for community detection in the stochastic block model. The threshold is defined as $(\frac{2log(q-1)}{\lambda^2(q-1)}, \frac{2(1+\mathcal{O}(1/logq))}{\lambda})$. q is the community number, and $\lambda = \mathcal{O}(1/q)$. There will be no optimal communities for graph structure if the average node degree falls out of the threshold boundary. [34] is another theory paper which offers two spectral algorithms to solve sparse graph community detection model. Derived from Grothendieck's inequality, [35] proves random graph semidefinite optimization consistency and illustrates its proofs in sparse graph community detection.

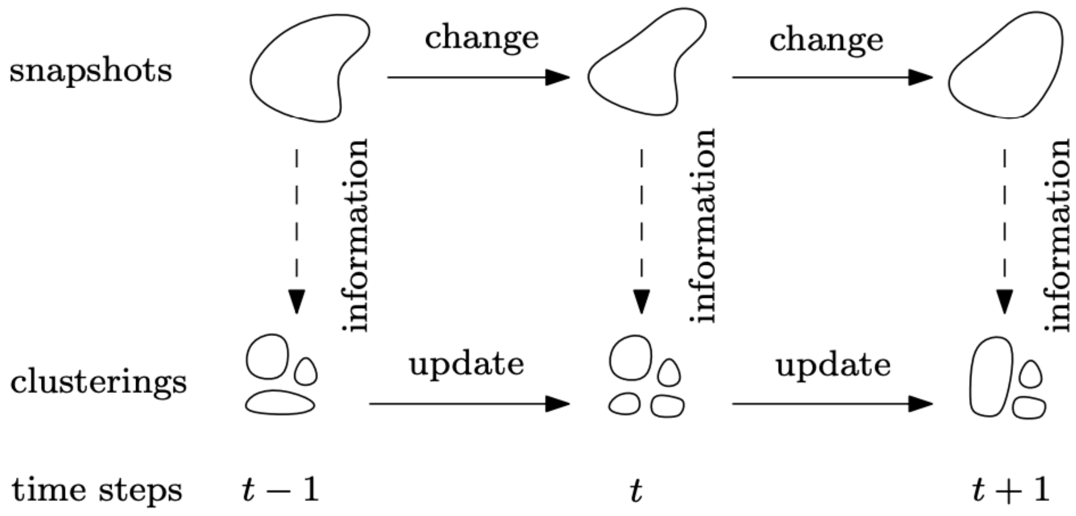


Figure 2.2: Dynamic community detection conventional steps. Figure is contributed from [36].

2.1.3 Dynamic Graph

Dynamic graph, also named as temporal graph in this paper, refers to a particular type of graphs which can be dynamically changed. This graph characteristic will lead to node community evolution, which is usually happened in social media or other real-world scenarios where the user profiles and interactions are updated frequently. [36] summarizes a set of relevant papers about the metrics, tasks and datasets. It especially lists a number of papers focusing online clustering. Figure 2.2 illustrates a typical framework of dynamic community detection.

[37] is a classic study about dynamic community detection. It aims to learn a community matrix C which best depicts the original graph adjacency matrix in a weighted form across all timestamps:

$$\min_{Y} \sum_{t} \sum_{i,j} |A_{ij} - \frac{N_i N_j}{2|E|}||Y_{ij} - A_{ij}| \tag{2.7}$$

where $Y = CC^T$.

DYNMOGA model [38] is a genetic based approach to detect smooth communities with multi-object optimization. It considers the accuracy of the generated communities in the temporal graph at each timestamp. Meanwhile it also tries to minimize the transformation cost between two community partition results in consecutive timestamps.

[39] combines two statistical models: a static model for individual graph snapshot in each timestamp and a temporal model to track the evolution of states. It proposes a prior & posterior stochastic block model, in which the previously learned parameters in the last snapshot are used to optimize the current community memberships via extended Kalman filter (EKF). Using similar approaches, SBTM model [40] leverages hidden Markov assumption in original SBM in which the presence/absence of an edge between two nodes will directly influence whether it will appear again in the next timestamp.

NEIWalk model [41] is a random walk method applied on dynamic content-based graphs by tracking the evolution of both graph structure and node/edge content. In detail, the model transforms the original graph into a Node-Edge Interaction (NEI) graph which contains two types of nodes and three types of edges. In NEI graph, the two node types are the node and edge in the original content-based graph. And the three edge types are the calculated structural similarity, node content similarity and edge content similarity. The dynamics in the original content-based graph is reflected to the structure evolution in the NEI graph. To detect communities in each timestamp, a random walk based model is proposed to involve a transition probability matrix between each node/edge type and thereafter detect communities in the NMI graph.

[42] also detects dynamic communities in content-based graphs by incorporating graph structure, content and temporal analysis in a shared matrix factorization model. In each timestamp, it considers to learn three following matrices in a holistic way: a matrix U_t at timestamp t based on both structural and content information, a global matrix S based on structural information and another global matrix W based on content information across all timestamps. The objective function uses these three matrices to best estimate the original adjacency matrix A and given content matrix C:

$$\operatorname{argmin} \sum_t ||A_t - U_t S^T||^2 + \beta \sum_t ||C_t - U_t W^T||^2 + \lambda(||S||^2 + ||W||^2 + \sum_t ||U_t||^2) \tag{2.8}$$

The first term aims to estimate A, the second term aims to estimate C_t at each timestamp t, and the third term is the regularization term to avoid bias and overfitting.

[43] focuses on dynamic community detection in graph streams by defining a Local Weighted-Edge-based Pattern (LWEP), which is a subset of node densely connected in local regions. The whole approach is divided into online and offline steps. In the online step, for each node in timestamp t, a top-k neighbor node list (with largest edge weights) and a top-k candidate node list (with highest burst activities) are extracted to preserve graph main structure. In the offline step, the LWEPs are created in each timestamp from online results. A straightforward two-step approach first cleans the generated LWEPs and then uses a breadth first search (BFS) method to figure out dynamic community structures.

[44] proposes a nonparametric multi-group membership model for dynamic graphs. Within its model, a community can be tracked as either active or inactive, which is assumed to follows Poisson distribution. In timestamp t, the dynamic joining or leaving of a node towards a community is estimated by a Markov chain process considering its previous status in the community. The estimated parameters follow Beta distribution. The community structure can be used to estimate the probability of edge affinities in the original graph. Finally a MCMC method is used to optimize all the to-be-learned parameters under all assumed distributions.

[45] is a theoretical paper which argues a threshold generated from community strength and dynamic change rate. It claims that there would be no effective methods if the dynamic graph has a score below the threshold. Further more, it mathematically proves belief propagation model and spectral clustering are able to detect reliable communities if the graph structure is above the threshold.

[46] is another statistical work which assumes evolved node-community memberships following Gamma distribution. The communities also follows a Gamma distribution. The edges between nodes follow Bernoulli distribution and their weights follow Poisson distribution. The whole process optimizes all aforementioned distributions through a MCMC process. Similarly, [47] uses Bayesian Markov Chain to generate sequential edges appeared in the graph without specified time

windows.

[48] introduces a new metric, stability, to assess community partitions. The stability of a community partition is defined as:

$$r(t, C) = \min_{0 \leq s \leq t} trace[C^T(\Pi M^t - \pi^T\pi)C] \tag{2.9}$$

where M is the node transition matrix, π is the normalized degree vector over all nodes and$\Pi = diag(\pi)$ is the corresponding diagonal matrix. C is the community matrix. The partition with maximized stability may not be the best one in current timestamp. But its community structure should be with good coherence in a relative long time period.

In [49], the time-evolving graphs can be represented as a set of sequential adjacency matrices, which can be integrated as a three-way tensor $\mathcal{T}$. The conventional matrix factorization can be thereafter applied to the tensor by finding three low dimensional matrices A, B, C to best estimate $\mathcal{T}$. A and B provides the graph community structure and C represents the temporal activity of communities. It is a PARAFAC decomposition problem to solve.

$$\min_{A,B,C} ||\mathcal{T} - A, B, C||_F^2 \tag{2.10}$$

[50] focuses on dynamic multi-mode graphs, which is a particular type of heterogeneous graph where the node types and edge connections are constructed from different modes such as text and videos. It uses a matrix factorization approach to learn node community matrix in each timestamp by estimating the interaction between each pairwise modes. And a side effect from snapshot graph in the last timestamp are also considered in the learning process.

2.1.4 Large Graph

Large scale graphs are ubiquitous in various domains. Especially in recent years, with the increasing computation capability, more and more researches focus on how to deal with large graph community detection. A well-known survey paper [7] introduces a number of methods focusing on both overlapping and disjoint community detection as well as many frequently used metrics and datasets for evaluation.

TopGC model[51] proposes a linear-time searching algorithm to detect the best community instead of the entire graph communities. It uses the idea derived from Locality Sensitive Hashing (LSH) to index nodes and find the communities with largest cluster scores (related to the community size and number of within-community edges). [52] is another local searching model which leverages random walks to detect the best community for each given node, and runs in approximately linear time.

[53] introduces a novel model to deal with large graphs by sparsifying graphs and removing unimportant edges to maintain the main graph structure. Once the graph size is reduced, the running time of most methods will be shortened as their complexity are always related to the graph size. To prune the graph in an efficient manner, a local sparsification algorithm is introduced to maintain the top ranked edges (edge score is calculated by the Jaccard Similarity of current node and the other endpoint node) per node. A further threshold is set up to ensure the pruned graph is still connected.

[54] formulates a novel problem, community kernel detection, which contains two tasks: find the most influential users and detect their community structure. It proposes two separate algorithms, GREEDY and WEBA, both to solve the two tasks in a unified framework. In the GREEDY method, it firstly initializes a set of seed nodes as kernel communities. Then these communities expands by merging nodes with closest connections. In the WEBA method, the model learns a weight vector for each node to represent its belonging score to each community. The weight vectors are learned by maximizing the inner product of the vector of all pairwise connected nodes.

GEM model [55] is proposed to cluster large-scale social networks. It consists of three steps: firstly it extracts the main skeleton from the original graph, and then detects the skeleton graph communities via a k-means method, in the last step the calculated result is propagated back to refine the communities of entire graph. In detail, for graph extraction, all nodes whose degree are above a threshold are selected and their edges are kept. Then a weighted k-means with refined seeding strategy is applied to cluster the filtered skeleton graph. In the end, the remained nodes in the original graph are assigned to the generated communities through a breadth first search (BFS) approach.

[56] gives a novel definition of a community in which all nodes are closer to the current community centers than to the rest communities. After that, two new algorithms (Cores-Aware and Cores-Unaware algorithm) are proposed to detect communities in linear time. The Cores-Aware algorithm is leveraged under the condition that community cores are known. Then a breadth first search (BFS) approach is applied to calculate other node distances towards all cores and assign their communities. In the Cores-Unaware algorithm, their is no prior-known community cores.

ESCG model [57] repeatedly generates supernodes with coarser resolution to replace the original nodes in bipartite graph, which reduces the original graph size and accelerate the running speed. In detail, the model first randomly picks up a number of seed nodes as individual supernodes. Then a BFS approach is taken to allocate the rest nodes to the seed nodes based on their shortest paths. In the end, a standard spectral clustering method is leveraged on the reduced graph to detect communities.

[58] proposes a mixed global and local method to bump up the efficiency via a defined k-path edge centrality as a substitution of the original edge betweeness centrality. Each edge importance score is calculated via random walks, which maps nodes to a latent space and calculates all pairwise distances for final community detection. The k-path edge centrality is defined as:

$$L^k(e_{ij}) = \sum_{v\ V} Pr(e, v) \tag{2.11}$$

It calculates the sum of probabilities that a k-step random walk passes edge e_{ij} which is originally started as node v. And the distance of pairwise nodes can be thereafter calculated as:

$$d_{ij} = 1 - \sqrt{\sum_{v \in V} \frac{(L^k(e_{vi}) - L^k(e_{vj}))^2}{d(v)}} \tag{2.12}$$

The Louvain method is finally applied on the all pairwise node distances to calculate their communities.

[59] deals with large probabilistic graphs in which each edge is associated with a probability to appear or not appear. It proposes a simple model to generate a cluster graph C which has the least edit distance with the original graph G. And the disjoint subgraphs in the cluster graph C naturally form the communities.

SCD model [60] is a scalable approach to detect communities by maximizing the Weighted Community Clustering (WCC) metric. It also allows to run in parallel, which achieves a two-order magnitude acceleration in terms of running speed. WCC is defined based on the intuition that nodes should have a higher chance to form triangles with in-community nodes compared with the out-community nodes. Therefore, for a given node v, its WCC score in a community c is calculated as:

$$WCC(v,c) = \begin{cases} \frac{t(v,c)}{t(v,V)} \cdot \frac{vt(v,V)}{|c-v|+vt(x,V-c)} & \text{if t(v,V)} \neq 0 \\ 0 & \text{if t(v,V) = 0} \end{cases} \tag{2.13}$$

$t(v,c)$ and $t(v,V)$ refer to the number of triangles containing node v in the community c or the whole graph nodes. $vt(v,V)$ refers to the number of nodes that forms close triangles with current node v. The sum of all nodes' WCC scores across all communities is the final score to evaluate the fitness of current partition. The paper uses an iterative approach to learn the community partition with largest WCC score. [61] also takes advantage of closed triangles in a spectral clustering approach.

[62] aims to find overlapping communities in large-scale graphs. It defines a local spectra which uses a set of seed nodes (with known communities) as the dimensions to represent all other nodes through random walks. The local spectra representation hugely reduces the workload to decompose the adjacency matrix. And a refined k-means method helps to group nodes into overlapped communities.

DFuzzy model [63] leverages deep learning techniques in four steps. First, it uses a PageRank method to initialize the potential community centers. Second, PageRank based clustering encodes

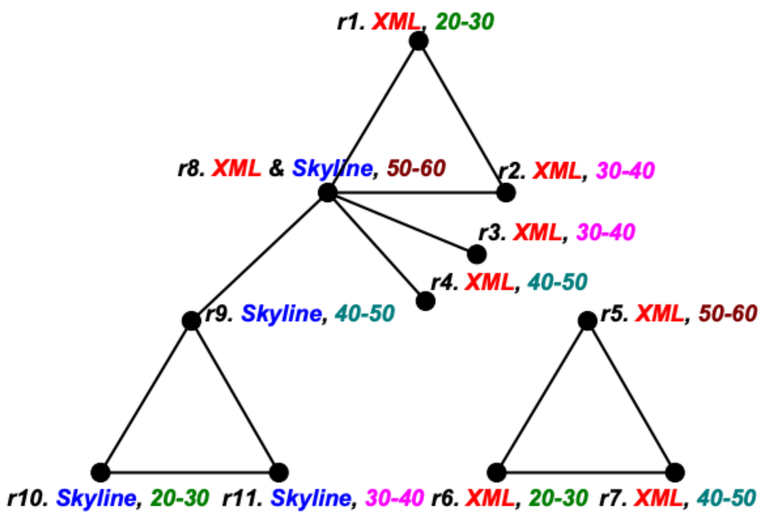


Figure 2.3: A co-author graph with two attributes "Topic" and "Age". Figure is contributed from [67].

node hidden representations by maximizing graph Modularities. Communities are detected in the decoding process by minimizing the reconstruction error in the unified encode-decoder framework.

Some other studies are relevant to large-scale graphs from different perspectives. [64] is a theory paper which introduces network lasso to enhance the optimization process. [65] is an empirical paper aims to identify under what scenarios that small communities are better/equal/worse than large communities. [66] offers a model selection strategy by considering graph minimum description length.

2.1.5 Attribute Graph

Intuitively, attribute graph is a type of graph in which nodes contains extra information. Figure 2.3 shows a typical attribute co-author graph with "Topic" and "Age" attributes. An attribute graph $G(V, E)$ is associated with an attribute collection $\Lambda = \{\lambda_1, ..., \lambda_m\}$ containing m different attributes. Each node v is associated with an attribute vector $[\lambda_1(v), ..., \lambda_m(v)]$ where $\lambda_1(v)$ is the attribute value of node v on attribute λ_m. A more complex attribute graph can involve attributes in edges as well with similar representations to node attributes. Bayesian approaches and nonnegative

References	**Main Idea**
[68, 69, 70, 71]	Bayesian generative & probabilistic model
[72, 73, 74]	Nonnegative matrix factorization
[75, 76]	Graph convolutional network
[77, 78, 67, 79]	Others

Table 2.1: The main ideas of mainly introduced attribute graph community detection approaches.

matrix factorization approaches are the two main tracks of methods, and their related works are summarized in Table 2.1.

Inc-Cluster model [77] uses random walk distance matrix instead of adjacency matrix to calculate node communities. The random walk distance matrix considers the random walk probability between each pair of nodes and their involved attributes. Thereafter, a Kmeans-like method is taken to initialize community centroids and run with the four following steps iteratively until convergence: first, it assigns nodes to the community with the nearest centroid random walk distance; second, it updates each community centroid from the newly formed communities; third, it adjusts attribute weights with an efficient incremental method; fourth, it re-computes the random walk distance matrix and starts the next iteration.

SCMAG model [79] aims to retrieve the dense communities from attribute graphs. It firstly leverages an entropy-based method to find dense subspaces. A random walk based method calculates the structural closeness and attribute similarity between communities together as a unified metric. A combining strategy is finally proposed based on the metric to merge subspaces together as communities.

[68] seamlessly captures both structural and attribute aspect information and develops a Bayesian probabilistic model to detect node communities. It leverages three matrices including adjacency matrix $X \sim Bernoulli(\cdot)$, attribute matrix $Y \sim Multinomial(\cdot)$ and node community label vector $Z \sim Multinomial(\cdot)$. The overall generative process firstly samples the community label for each node. Then given the community label and node itself, the model samples node attributes. In the end, given community labels for each pair of nodes, the model samples whether there is an edge between them. The overall idea is to reproduce the original graph by optimizing

these three generation probabilities.

CRM model [69] is also a probabilistic model to incorporate all user information from social networks. The overall approach aims to generate three types of user information including community (each edge belongs to a community), role (each node can have multiple roles) and action (edge node can take multiple actions such as transferring a message). The three generative process are interacted with each other and optimized through an EM approach.

PAICAN [70] is a jointly model which learns partial node anomalies and communities via a statistical inference perspective. A node is a partial node anomaly when it is an outlier in at least one attribute. In the graph generative process, an edge is generated from different distribution assumption under three case of endpoint node information: both nodes are good, one endpoint node is good, and both nodes are anomaly. [71] is another model involving node semantics. Similarly, the overall process is a joint community $\rightarrow$ edge generation and topic $\rightarrow$ attribute generation model.

[78] aims to detect top k communities which can both group cohesively connected nodes together and concisely describe their attributes. It defines two types of errors to assess community partition fitness. The first type error is missing edges between nodes in the same community. The second type error is the edges existing between communities. The model calculate the scores of community given these two criteria. The idea of this paper is to generate concise queries from node attributes and selects top k best fit communities for these queries.

CESNA model [67] detects overlapping communities by considering both edge structure and node attributes. It is a scalable algorithm with linear running time. The intuition behind CESNA model is that node community memberships should be able to indicate node attributes. The probability that a node u contains attribute k is described as:

$$P_{uk} = \frac{1}{1 + exp(\sum_c W_{kc} \cdot F_{uc})} \tag{2.14}$$

where F_{uc} is the affiliation score that node u belongs to community c. Weight W_{kc} is the weight factor for attribute k towards community c.

[72] shows how edge content can be used to improve the effectiveness of community detection. The overall model is derived from matrix factorization. It considers to minimize both structural object and edge content object in a unified function $\mathcal{O}(E)$:

$$\mathcal{O}(E) = ||E^T E\Delta - \Gamma||_F^2 - \lambda tr(E^T LE) \tag{2.15}$$

The first part is the structural object to minimize, where Γ is a $|E| \cdot |V|$ matrix to encode edge and node underlying structure. $\Gamma_{ij} = 1$ if v_j is the start-or end-point of e_i. Otherwise $\Gamma_{ij} = 0$. E is a low dimensional matrix representation for all edges. Δ is a $|E| \cdot |V|$ matrix whose entry $\Delta_{ij} = \frac{1}{d(vj)}$ if v_j is the start-or end-point of e_i. The second part is the edge content object. $tr(\cdot)$ denotes the trace of relevant matrix. λ is a weighting factor. L is the normalized Laplacian matrix storing all pairwise edge content similarities. In the end, edge vectors are learned and used for community detection.

[73] aims to learn a community membership matrix U and community attribute matrix C through a nonnegative matrix factorization framework. The objective function in this model is defined as:

$$\underset{C\geq 0, U\geq 0}{\operatorname{argmin}} ||U - SC||_F^2 + \alpha \sum_{j=1}^{k} ||C(:,j)||_F^2 + \beta ||A - UU^T||_F^2 \tag{2.16}$$

where U_{ij} is the propensity of node i belongs to community j. C_{qr} is the propensity of that attribute q belongs to community r. S is the given node attribute matrix. The first term aims to estimate node community membership matrix from node attributes. The second term is an add-on bias. The third term is to use community membership matrix to estimate the original adjacency matrix A.

[74] incorporates node content into a nonnegative matrix factorization approachc and use a weight matrix C to control the impact of the attribute information for community detection. The objective function it aims to optimize is as:

$$\underset{X\geq 0, Y\geq 0}{\operatorname{argmin}} \alpha ||A - XX^T||_F^2 + ||X - CY||_F^2 \tag{2.17}$$

where A is the adjacency matrix, X is the community membership matrix, Y is the community attribute matrix to describe how much likely the community can be described by each attribute. C is the node attribute matrix. Both X and Y are the matrices to be learned.

[75, 76] are both deep-learning based method to involve node attribute into graph convolution process and trained in an end-to-end manner.

2.1.6 Summary

In this section, I introduce five different types of graphs and a bunch of related research works. Some of the graph types are commonly seen in real-world scenarios such as sparse and large graphs. Effective and efficient models on such graphs can produce huge practical value to reduce online latency and required machine memories. Researches in attribute graph is becoming more popular because it can involve extra information to support graph partition. The extra information is always with meaningful semantics, clear format and easy to be obtained. Besides the type of graphs introduced in this section, there are also other types of works, such as signed graph and hypergraph, which are all interesting topics and deserves more future exploration.

2.2 Tasks

In this section, research works about five popular community detection tasks are summarized. The five selected tasks are independent with each other and tackle community detection problems from various perspectives.

2.2.1 Overlapping Community Detection

Overlapping community detection is one of the most fundamental topics and many relevant studies are published. By definition, a node v can belong to multiple communities simultaneously so as to cause the overlaps between communities. Figure 2.4 visualizes the community structure under overlapping circumstance. The nodes marked in color yellow are affiliated with both community A and B. Table 2.2 categorizes and summarizes the focus of each mentioned study for overlapping

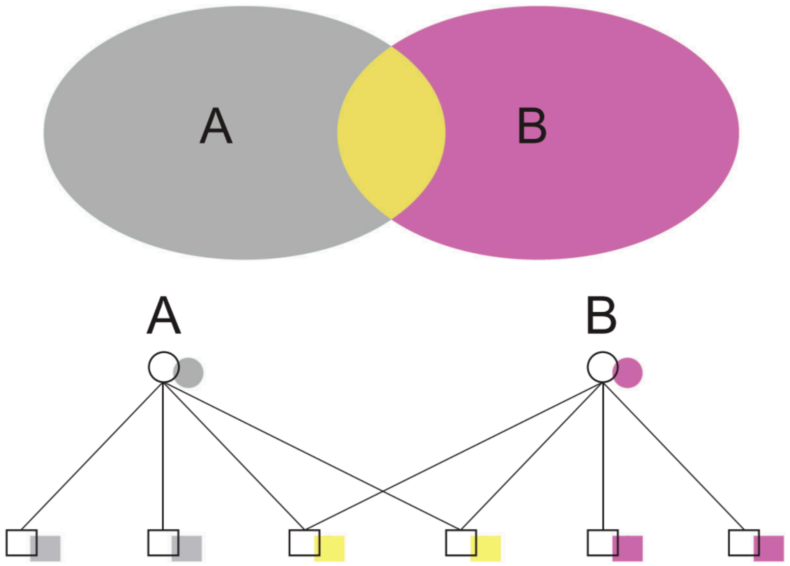


Figure 2.4: Overlapping Community Structure where nodes can belong to multiple communities. The figure is contributed from [80].

References	**Main Idea**
[81, 82, 83, 23, 84]	Local search and seed expansion
[80, 85, 86, 87, 88]	Nonnegative matrix factorization
[89, 90]	Bayesian, generative model
[91]	Affiliation network

Table 2.2: The main ideas of mainly introduced overlapping community detection approaches.

community detection.

[11] is a survey paper published in 2014. Although the methods introduced within it are sort of dated, it still offers covers several main track methodologies (node seed and local expansion, clique expansion, and label propagation) and particular scenarios (link clustering and dynamic graphs). It draws a conclusion that there is no universal method to deal with all types of graphs with different characteristics in sparsity, degree distribution, and overlap percentage among communities. In the end, it also points out two essential questions to be solved for later researchers, which is "when to apply overlapping methods and how significant the overlapping is". [10] is another authentic survey paper which reviews a set of methods, benchmarks and evaluation metrics. Besides

that, it also reviews papers using stochastic block model or density based models. To study the performance differences among models, it introduces several benchmark graphs in which node ground truth community is known, i.e. LFR benchmark graph[1]. A set of evaluation metrics including Normalized Mutual Information (NMI) and Omega index are introduced as well. Meanwhile, empirical studies are applied and evaluated on all aforementioned methods using different benchmark graphs. In the end, the paper discovers the sensitivities of different models in sparse graphs appeared in real-world social networks.

DEMON model [81] unveils its overlapping communities with a local-first approach by grouping ego neighbor nodes into the same clusters and finally merges the local communities into a global collection. In the local-first grouping process, DEMON applies a EgoMinusEgo function to first extract ego-based subgraphs for each node v where the node set is node v and all its neighbor nodes $\mathcal{N}(v)$ and the edge set is all graph edges between the selected nodes (v and $\mathcal{N}(v)$). After that, each ego-based subgraph will remove the node v and all its associated edges to achieve an EgoMinusEgo subgraph. An label propagation community detection method is subsequently applied on each EgoMinusEgo subgraph and select a set of largest clusters to best cover the entire graph. In the end, a merging process is applied on the previously generated clusters according to their node similarities and construct the final community partition result. LOSP model [92] is another method to explore local neighborhood structures of each node. It defines a Local Spectral Subspace using the first d eigenvectors from the normalized adjacency matrix. In each potential community c, a set of seed nodes are given, and an iterative process is applied with the help of seed nodes and the Local Spectral Subspace to rank the top N_c nodes with highest random walk probabilities to appear in the current community. Those nodes are finally regarded as other latent members in each community c.

LOSP model offers a great insight to use seed node expansion to detect overlapped communities. Inspired by LOSP model, [82, 83] propose a four-phase model including filtering, seeding, seed set expansion, and propagation. The natural of the model first filters out the regions of the

[1]https://sites.google.com/site/andrealancichinetti/files

graph which don't involve overlapping structure. A seeding strategy inspired by Kmeans selects a set of seed nodes with small Conductance relationship. A seed set expansion approach is further applied by taking advantage of personalized PageRank model to construct raw communities near each seed node. In the last propagation step, the raw communities are expanded again to the regions previously removed in the filtering phase. Similarly, [93] also designs a new seeding strategy and expands a set of nodes from seed nodes via a personalized PageRank model. Thereafter it develops a model on the expanded nodes to detect overlapped communities. The seed nodes are the nodes in the maximum spanning tree of the original graph. Personalized PageRank model helps to expand the seed nodes and merge them into raw community structure by maximizing modularity score of the graph. The last optimization step is to merge communities when the shared node ratio of two communities is above an overlapping threshold.

OCD-HSN model [23] contains a seed selection step as well as community initialization and expansion step to group nodes into clusters in an efficient manner. It claims to be the first research work for overlapping community detection in heterogeneous graph containing both undirected and directed edges. Based on user semantic interests and social connections, this study constructs a heterogeneous graph to hold all types of user profiles. A set of seed nodes are selected based on their centrality and conductance in the graph. The neighbor nodes of the seed nodes naturally construct overlapped communities. Later on, a fitness evaluation process is leveraged to add or remove nodes based on the node connectivity in each community.

Structural centers are defined and utilized in [84] to support local expansion for overlapping community detection, which are the nodes that have high node degrees and are also far away from other structural centers.

By natural, nonnegative matrix factorization can solve overlapping community detection problem as it can directly learn low dimensional node representation matrix from the graph matrices such as adjacency matrix. Each column in the reduced node represention matrix represents an eigenvector, which also can be regarded as a community. Therefore, the learned node representation can be regarded as the node affiliation distribution over communities. BIGCLAM model [80] is a classic overlapping community detection model which can take care of large-scale graphs. In its model, F represents a nonnegative matrix where F_{uc} is the affiliation score of node $u \in V$ in community $c \in C$. Given an unlabeled and undirected network $G(V, E)$, BIGCLAM aims to re-generate all graph edges using node's community affiliation distribution. It aims to maximize the probability of node u and v if there is an edge between them and minimize the probability if there is no edge. Therefore, the objective function $l(F)$ to maximize turns to be:

$$l(F) = \sum_{(u,v)\in E} log(1 - exp(-F_u F_v^T)) - \sum_{(u,v)\notin E} F_u F_v^T \tag{2.18}$$

The number of communities can also be determined in the model through an empirical test.

PNMF model [85] is also a nonnegative matrix factorization model. Unlike conventional matrix factorization models to directly decompose the original adjacency matrix, PNMF considers the situation that a node prefers its neighbor nodes. Therefore, instead of re-generating the edges between two nodes, this paper considers node triplet (i, j, k). $j >_i k$ denotes that $j \in \mathcal{N}(i)$ and $k \notin \mathcal{N}(i)$ so that node i prefers j to k. And the overall goal of this approach is to maximize this preference likelihood that neighbor nodes should be preferred than non-neighbor nodes:

$$\begin{aligned} l(F) &= \sum_{(i,j,k)\in S} log(p(j >_i k|F)) - \lambda \cdot reg(F) \\ &= \sum_{(i,j,k)\in S} log(\sigma(F_i F_j^T - F_i F_k^T)) - \lambda \cdot reg(F) \end{aligned} \tag{2.19}$$

where S is pre-sampled node triplet collection which satisfies the node preferences. $reg(F)$ is the regularization term added to avoid overfitting and λ is the regularization weight.

HNMF [86] is another nonnegative matrix factorization model which considers both-sided relationships between edges and communities. From the community-to-edge perspective, it takes advantages of PNMF model to maximize the log likelihood of node connection preferences in all sampled node triplets. From the edge-to-community perspective, it uses negative sampling strategy in a skip-gram model to maximize the probability if two nodes are neighbor nodes and minimize the probability if they are not. The final loss is the loss sum from both PNMF and the negative sampling, which is aimed to be maximized in the joint training process.

NRATIO model [87] generates a vertex neighborhood ratio matrix to substitute original adjacency matrix or Laplacian matrix. This matrix refines the graph adjacency matrix where two nodes are connected only if their common neighbor nodes surpasses the average number of neighbors in all pairwise nodes. In the next step, nonnegative matrix factorization is applied on the refined matrix to learn a community distribution over each node. As the vertex neighborhood ratio matrix reduces the influence of unrelated nodes in community structures, the number of data points in the matrix are significantly reduced, which fastens the running speed.

[88] first describes a stochastic block model to accommodate both node and edge communities, and then uses conductance measurement to fine-tune the learned node community distribution. One outstanding point of this paper is that the model considers edge communities. By definition, in the adjacency matrix A, $a_{ij} = 1$ if node v_i and node v_j are connected through an edge. And, in bipartite graph matrix B, $b_{ij} = 1$ if node v_i and edge e_j are directly linked each other. The paper aims to learn a node community affiliation matrix H where h_{ij} denotes the propensity of node v_i belonging to community c_j and an edge community affiliation matrix W where w_{ij} denotes the propensity of edge e_i belonging to community c_j. In the end, to use the node/edge community information (W and H) to re-construct the two matrices A and B, the objective function is defined as follows:

$$\mathcal{O}(H, W) = ||A - HH^T||_F^2 + \lambda||B - WH^T||_F^2 \tag{2.20}$$

where $||\cdot||_F$ is the Frobenius norm, H and W have to be nonnegative matrices.

AGM model [91] is a preliminary study using affiliation network to re-generate the original social network. Given only the overlapping community affiliation of each node, it learns the reproduction of original links between nodes in order to construct the social network. The edge generation probability is defined as:

$$p(u,v) = 1 - \prod_{c \in C_{uv}} (1 - p_c) \tag{2.21}$$

where C_{uv} are a set of shared communities for node v and u. p_k refers to the probability of an edge forming between two nodes in the community c.

[89] uses a mixture membership stochastic block model to learn node overlapping communities. The overall framework is a Bayesian approach. It assumes the overlapped community memberships of each node v_i is associated with a Dirichlet distribution θ_i. For each pair of node v_i and v_j, it draws a community indicator $z_{i\to j}$ from node v_i's community membership θ_i and then draws a community indicator $z_{i\leftarrow j}$ from node v_i's community membership θ_j. Each indicator points to one of the $|C|$ communities. Finally, it draws an edge between two nodes with the generation probability:

$$p(y_{ij} = 1 | z_{i\to j}, z_{i\leftarrow j}) = \begin{cases} z_{i\to j}, & z_{i\to j} = z_{i\leftarrow j} \\ \epsilon, & z_{i\to j} \neq z_{i\leftarrow j} \end{cases} \tag{2.22}$$

The whole process is optimized and all nodes' θ are learned by maximizing the edge generation probability $p(y_{ij} = 1)$from the actual graphs. ϵ is a small constant. The parameter $\beta_{z_{i\to j}}$ is the density of community $z_{i\to j}$.

Similarly, [90] also introduces a Bayesian based generative model. It assumes a Poisson distribution for node community distribution and is optimized using a Expectation-Maximization (EM) approach. Node communities are selected according to its learned community distribution and related community conductance in the end.

2.2.2 Number of Communities

Among all sorts of community detection models, only few of them can automatically determine the number of communities given the model nature. Most of them need to specify the community number in advance before applying their main processes. Therefore it is still an open question about how to choose the exact community number. Many in-depth researches draw their conclusions either from empirical studies or mathematical proofs. In this section, several works particularly exploring this research question are introduced and discussed, which offers general insights for potential community number selection.

[94] proposes a statistical model using Bayesian inference analysis. It finds the correct number of communities by maximizing the integrated likelihood of graph structure and observed community partition using a generative model. In detail, given the prior knowledge of graph structure (adjacency matrix A) and community partition C, it uses Markov chain Monte Carlo (MCMC) importance sampling to calculate the posterior distribution of community number $|C|$.

[95] develops an efficient model to study community numbers based on graph spectral properties. It statistically proves the number of positive and most informative eigenvalues as the estimated number of communities on five spectral clustering methods which use either the non-backtracking matrix or the Bethe Hessian matrix.

Moreover, two motivations happened in stochastic block models urge the exploration of community numbers. First, edges are not necessarily independent if only the communities of their end-point nodes are given. Second, the loss of precision occurred in spectral clustering is inevitable. Therefore, through a set of rigorous proofs, [96] proposes a composite likelihood BIC (CL-BIC) model to handle the community number selection happened in stochastic block model (SBM), degree-corrected block model (DCBM) and mixture-membership block model (MMB).

For other works relevant to community number selection, [97, 98] both leverage the leave-one-out cross-validation to determine the community number by optimizing the edge prediction error and Bethe free energy in stochastic block model. [99] conducts a comparative analysis on various community detection models (stochastic block model, greedy methods, statistical inference models and spectral methods) to track the changes of assessment criteria (modularity, map equation, Bethe free energy and prediction error and isolated eigenvalues) associated with various of community numbers.

References	Main Idea
[100, 101, 102, 103, 104]	Return the densest subgraph containing all query nodes
[105, 106, 107, 108, 109, 110, 111]	Return top k most relevant/influential communities with or without query nodes

Table 2.3: The main ideas of community search approaches.

2.2.3 Community Search

Instead of generating communities for all nodes in the graph, community search based approaches aim to find the top relevant subgraphs or communities that match particular purposes, which can be regarded as a mixture of community detection and information retrieval. Currently, there are two main types of search scenarios. One is that given a set of query nodes, return the densest subgraph containing query nodes. The other is given a set of query nodes, return their top k most relevant communities. Relevant works are summarized by main ideas in Table 2.3.

[100] is a very classic paper of community search, which is defined as given a graph $G(V, E)$ and a set of query nodes, seeking a a subgraph of G which contains all the query nodes and meanwhile is densely connected. The nodes with trivial degrees and far away from query nodes are filtered out from the community. The final goal of this paper is to maximize the minimum degree of a subgraph containing all query nodes. A greedy solution is proposed by deleting each node with minimum degree at each step. The iterative process terminates either at least one of the query node reaches to the minimum degree or is no longer connected with the rest of nodes. Further two heuristic approaches are extended to detect the community with an upper bound size restriction.

[101] is a subsequent work of the previous one, which aims to detect the best community for a given node. It formulates and solves two community search problems: communitysearch with a threshold constraint (CST) problem and community search with a maximality constraint (CSM) problem. Two approaches including global search and local search are proposed where the second approach is more efficient than the first one because of reducing searching space significantly.

[102] is another follow-up work of [100]. It improves the model efficiency and accuracy as one

limitation of the previous work is that it tends to find quite large solutions, which negatively affects the detected community accuracy. Unlike the [100] in which the community size is constrained, it aims to detect the smallest-size community among all optimal ones. In detail, it computes the core decomposition and organizes the retrieved k-cores (maximal subgraphs where each nodes are connected at least k other nodes in a subgraph). In the end, a greedy-connection approach is leveraged to use a local search method to reduce the potential subgraph size (greedy step) and a Steiner Tree-based strategy to find the minimum number of nodes that make all query nodes connected.

Given a query node, [103] formulates a query biased densest connected subgraph (QDC) problem which aims to find the densest subgraphs that contains the query nodes and is also internally connected. A densest subgraph can be regarded as a local community in this paper. To reduce a free rider effect which involves irrelevant nodes into densest subgraphs, a random walk based weighting scheme is proposed to set higher penalties to these irrelevant nodes when they are included in the subgraphs/communities.

[104] solves a closest truss community (CTC) problem which finds a connected k-truss subgraph with the largest k and minimum graph diameter. By its definition, a k-truss subgraph requires each edge is contained in at least $k-2$ triangles in the subgraph. A greedy algorithm is proposed to first find a raw CTC that contains all query nodes, then iteratively removes the furthest nodes in the CTC to finally achieve the optimized subgraph.

[105] introduces a partial clustering model to extract the most dense subgraphs, meanwhile it does not require to determine the number of subgraphs in advance. The model is inspired by the matrix blocking problem to reorder the adjacency matrix A and to find the dense diagonal blocks as extracted subgraphs. It considers three types of graph scenarios including undirected graph, directed graph and bipartite graph. By calculating the pairwise columns in the adjacency matrix, it obtains the similarities between all pairs of nodes, which is stored as matrix M. The final goal turns to reorder the rows and columns of M so that inside each nontrivial diagonal block of matrix M, there exists at least one datapoint larger than a threshold for each row and column. It means

that for each node in the diagonal block, there exists at least one other node densely connected with it. To accelerate the running speed, a hierarchical optimization approach is proposed in a top-down manner. The large generated diagonal blocks are finally regarded as extracted dense graphs.

[106] aims to retrieve top k locally densest subgraphs (LDS) which are particularly defined in the paper. LDS is defined based on density (nodes within the LDS are densely connected) and compactness (removing a node in LDS will lose significant number of edges within it). With solid proofs, LDS should satisfy a set of structural properties in compactness, cohesive and disjoint property. The original optimization approach is based on greedy algorithm, which runs in polynomial time. To reduce this running time, three optimization strategies are applied including pruning invalid nodes, optimization in finding dense subgraphs and optimization in verifying whether the dense graphs are eligible LDS.

OSLOM model [107] finds statistically significant communities through local optimization of a fitness function (significance score) which considers single cluster analysis and full network analysis. In single cluster analysis, the significance of a community is defined as the probability of finding the community in a random null model. In full network analysis, an iterative approach is leveraged to optimize the whole graph-level significance score by grouping nodes into proper communities. The approach first adds external nodes to existing subgraphs based on the generative probability. After that non-significant nodes are removed from the updated subgraphs.

[108] regards a node as a query in the graph, and retrieves its k-truss communities through a designed compact and elegant index structure, which makes the overall model scalable with linear cost to serve online community search tasks. On top of k-truss subgraph, k-truss community is defined with an extra edge connectivity constraint to ensure its node connectivity, which is any two edges in a k-truss community should either belong to the same triangle, or are reachable from each other through a series of adjacent triangles. To accelerate the running speed, a simple k-truss index are developed using an existing truss composition algorithm. Further an enhanced TCP-index is proposed to avoid two drawbacks from previous index: unnecessary access of dis-qualified edges and repeated access of qualified edges.

Having the same research goal, [109] takes edge weights into account to support better k-truss community detection for query nodes. It firstly utilizes a Breadth first search (BFS) method by removing unimportant edges iteratively. However this BFS approach suffers high computational cost which can not deal with large scale graph. To tackle this problem, it designs a KEP-index (Key Edges Preserved Index) which stores all possible k-truss community candidates in a space-efficient index structure. In the index construction process, communities are decomposed (edges are removed) in a hierarchical manner and the related key edges to reconstruct the hierarchical community tree are stored in the tree-shaped index.

[110] takes advantages of random walks to detect local graph communities. Starting from a set of seed nodes with given community memberships, the method labels each seed nodes with same/different colors according to its community. And a color-based random walk is applied to propagate colors across nodes to detect all other nodes communities. To store the color of each node, K transition matrices are maintained where K refers to the number of distinct colors in the seed nodes. Throughout an iterative propagation process, the color of each node can be learned and updated by its last-iteration color and the propagated colors from its neighbor nodes. The final color of the nodes are their deterministic community labels.

Derived from k-core, [111] proposes an online search algorithm to find the top k-influential communities in linear time through a linear-space index structure. A k-influential community means each node weight within should be at least k. The paper introduces a basic algorithm, a depth first search (DFS) and a final index based algorithm. The index based algorithm is able to handle large networks and runs in linear time. It uses a tree/forest index structure to hold all k-influential communities and utilizes a DFS function to iteratively add/remove nodes for index organization until all possible communities are scanned.

2.2.4 Community Detection Enhancement

Enhancing community detection performance is also an essential task as it provides extra in-depths supports to explore how to improve model performance. As there is no main trend in this research topic, different types of approaches are introduced here with dispersive solutions, including assigning edge weights, removing unimportant nodes, or involving external information, etc.

[112] assigns proper edges weights on unweighted graphs to circumvent resolution limit & extreme degeneracy problems and in the end enhances the performance of modularity based methods. modularity has been proved to suffer a resolution limit problem that the community size is constrained. With solid proofs, applying a weighted modularity can increase the upper bound and decrease the lower bound of community size. It allows to detect both larger and smaller communities using weighted modularity methods. Extreme degeneracy problem means that it goes more and more difficult to find the optimal community partition when the community size is large. And a proper weighting schema is able to strength the intra-community edges and weaken the rest to better clarify the community boundary. The weighting schema is defined as:

$$W_{ij} = \begin{cases} \frac{b_{ij}^{-\alpha} \cdot C_{ij}^{\beta}}{\sum_{k,m;k \neq m} b_{km}^{-\alpha} \cdot C_{km}^{\beta}}, & A_{ij} = 1 \\ 0, & A_{ij} = 0 \end{cases} \tag{2.23}$$

where α, β are positive numbers, b_{ij} is the edge betweenness score and C_{ij} is the common neighbor ratio between node v_i and v_j. [113] proposes a similar weighting strategy that edges are weighted according to their betweeness centrality score. The proposed WERW-Kpath approximately estimates edge betweenness score as the traversed probability of ρ random walks with at most k steps started from random nodes. This approximation reduces the NP hard edge weighting problem to a situation with worst time complexity as $\mathcal{O}(k|E|)$.

[114] shows a network preprocessing step can helps to alleviate the resolution limit problem for modularity based algorithms. It assigns random walks on graphs. By tracking these random walk trajectories, the similarity score of each pair of nodes are calculated as the updated weights in the original graph. The iterative steps run for several rounds and the modularity based community detection is applied in the last phase of graph.

[115] raises a point that hub nodes which are connected with many other nodes can disturb the community structure. Therefore, it proposes a degree-targeted node removal approach to reduce

such noise. In detail, it exhaustively searches through each node and introduces two types of approaches. First, the nodes with highest degrees are regarded as noisy nodes to remove. Second, the nodes whose removal causes the largest increase in modularity are regarded as noisy nodes to remove.

[116] first calculates the structural similarity of each pair of nodes based on their degrees. A strong constraint matrix and a weak constraint matrix are both derived from node pairwise similarities. Further on, these matrices are incorporated into a stochastic block model to enhance the community detection. Intuitively, it aims to group each node in the same community with its high-similarity nodes and separate with low-similarity nodes in different communities. The parameters to be learned are optimized through a nonnegative matrix factorization approach.

[117] designs a semi-supervised algorithm to incorporate prior knowledge to guide the detection process. The prior knowledge utilized in the study contains must-link and cannot-link. It simply applies this information in the adjacency matrix A: If node v_i and v_j have a must-link, the weight of $e_{ij} = \alpha$; If node v_i and v_j have a cannot-link, the weight of $e_{ij} = 0$. It further extends to another refined adjacency matrix but involving a third node v_k: If v_i and v_j have a must-link and v_i and v_k have a must-link, then v_j and v_k should also have a must-link (the friend of my friend is my friend); If v_i and v_j have a must-link and v_i and v_k have a cannot-link, then v_j and v_k should also have a cannot-link (the friend of my enemy is my enemy). Conventional community detection methods can be directly applied on the refined adjacency matrix to get the enhanced community partition.

[118] is the latest work which introduces two adversarial enhancement methods for community detection. One is called AE-GA, which is a genetic based adversarial algorithm to determine optimal edge modification and rewire community connections. Each chromosome in the genetic algorithm is a set of edges to be added or deleted in the graph. The fitness function is a variant of modularity:

$$f = \frac{\mathcal{Q}}{e^{|M_S - M_{real}|}} \tag{2.24}$$

where Q is current partition's modularity score. M_{real} is the number of the ground-truth communities in graph G. M_S is the number of communities detected by an particular method S. Another proposed model is called AE-VS model, which enhance the graph structure using node similarities. In detail, it rewires a graph by adding/deleting edges with a consideration of multiple similarity metrics and aggregates scattered communities to a better core community.

EdMot model [119] is a Motif-aware community detection. A motif-based hypergraph is constructed from the original graph to encode higher order node connections. Each of the top K largest connected components, with the most number of nodes, is individually partitioned into communities through modularity-based methods. All the potential edges (generated from each pair of nodes) within each community are strengthened to put back to the original graph for enhanced community detection.

[120] presents a way to automatically learn the resolution parameters in community detection to control the size of communities. It is formulated to optimize a parameter fitness function which generally measures how these parameters can solve a generalized community question through a set of solid proofs. [121] incorporates hierarchical concepts of node attributes into community detection. It detects node communities and simultaneously maintains the coherence of community label/context summarized from node attributes.

2.2.5 Semi-Supervised Community Detection

Conventional community detection approaches should be unsupervised and only leverage graph topological structures. However, unsupervised approaches are always weak in terms of model performance due to no-label guidance in the learning process. Therefore, to improve the model performance, partial information/constraints (must-links and cannot-links) are obtained to guide the learning process to the right direction, which forms the semi-supervised community detection task.

SNMF-SS model [122] combines symmetric nonnegative matrix factorization and a semi-supervised approach where domain knowledge is incorporated to guide the detected communities.

It provides two type of domain knowledge including must-links (W_{ML}) and cannot-links (W_{CL}). Each datapoint $w_{ij} \in W_{ML}$ means the edge between node v_i and v_j is given, and $w_{ij} \in W_{CL}$ means there should not be an edge between the two nodes. In the end, the paper aims to learn a low dimensional representation of nodes by optimizing the following objective function:

$$\underset{\hat{A} \geq 0, H \geq 0}{\operatorname{argmin}} ||\hat{A} - HH^T||^2 \tag{2.25}$$

where $\hat{A} = A - \alpha W_{ML} + \beta W_{CL}$ and A is the adjacency matrix. H is the low dimensional node matrix to be learned.

SSNMF model [123] involves must-links M as prior information into a nonnegative matrix factorization framework to alleviate the negative impacts from node degree heterogeneity. In this model, two nodes having a must-link should be restrictively grouped into the same community using the learned low dimensional node representation matrix from NMF approach. It turns to minimize the representation difference between the two nodes having must-links. And the final objective function to minimize turns to be the weighted sum of must-link restrictions and original NMF objective function:

$$\underset{\hat{A} \geq 0, H \geq 0}{\operatorname{argmin}} ||\hat{A} - HH^T||^2 + \lambda \sum_{ij} ||h_i - h_j|| M_{ij} \tag{2.26}$$

where H is the low dimensional node representation matrix and $h_i \in H$ refers to the node v_i's representation.

PCSEO-SS model [124] is able to detect false connections and conflicted connections. In detail, it utilizes a dissimilarity index metric which describes the probability of two nodes belong to the same community. Higher value refers to less same-community probability. Prior node pairwise constraints (must-links and cannot-links) are corrected by setting up a threshold on the dissimilarity index. Then an approach is given to maximize the modularity of the graph by removing extra-community edges iterative and in the end constructs the final community partition.

[125] also makes full use of node pairwise constraints including must-links and cannot-links

to extract high-quality communities. First, The two nodes in each cannot-link are forced to be grouped into different communities, which forms the skeleton of the overall community structure. Second, the communities containing nodes involved in must-links are merged into the same community without violating the existing skeleton community structure. After the nodes in must-links and cannot-links are processed, a greedy algorithm is applied to classify each of the remaining nodes into the existing raw communities. The nodes are selected and assigned to communities based on its similarity between each community and itself.

[126] proposes an iterative approach containing three steps. First, it applies nonnegative matrix factorization on the adjacency matrix to obtain the community structure. Second, for some pairs of nodes, their edges are uncertain but informative. The model designs a Connection Strategy by using human labeling on these edges to identify whether these edges should exist or not. Third, for these labeled new edges, a Disconnection Strategy is applied to highlight their importance weight, leading a result of graph topological reconstruction. The three steps are optimized iteratively and the final round result will be regarded as the final community partition.

[76] is the latest work by combining deep learning, Graph Convolutional Network (GCN), and statistical model, Markov Random Fields (MRF) to solve attribute graph community detection. The introduced end-to-end approach contains three layers. The first two layers are GCN layers which utilized both adjacency matrix A and attribute matrix X to learn the node community labels. The third layer is a MRF layer to refine the community partition result by enhancing node pairwise potential relationships.

WSCDSM model [127] contains two parts: A nonnegative matrix factorization approach with prior must-links to detect node communities and a semantic driven community detection via node content information. In the end, it integrates the two parts of the model into a unified framework and detects both topological and semantic communities. The training process combines the two individual objective functions together and uses a stochastic gradient descent method to optimize the two communities of each node.

For other types of approaches, [128] uses spin-glass model, [129] studies belief propagation and the stochastic block model, and [130] adds graph regularizations to penalize the dissimilarities between node pairs which should be in the same communities.

2.2.6 Summary

I formulate five different community detection tasks and introduce the best representative works of each task in the recent decade to demonstrate an overview. In fact, for each task, more and more research works having been published each year as the graph mining domain is with an increasing trend. Among the five tasks, overlapping community detection is the largest track which attracts the most researchers' attention for years. A rich number of papers are published regarding as this topic to the venues in computer science and physics domain. Semi-supervised approaches are also popular these years because empirical studies show that unsupervised learning approaches performs far worse than supervised ones. Moreover, beyond these five tasks, there are also many other tasks which focus on other perspectives towards community detection, such as Motif-based problems, math proofs, and edge community detection. In the future, more other interesting topics will be proposed, explored, and summarized.

2.3 Main Track Methods

The main track of methods in community detection is quite clear according the published years of milestone papers. Modularity based papers are in dominant position in early 2000s after the concept of modularity is defined by [131] and [132]. Spectral clustering methods start to raise in early 2010s, which aims to calculate eigenvectors from graph Laplacian matrix. [12] is a representative survey paper summarizes a series of relevant researches. Matrix factorization based approaches are also very popular in the similar time period to spectral clustering, as both types of approaches utilize matrix decomposition techniques. Stochastic block model is a type of statistical inference model which has a lot of variants in the recent decade. Recently with the rapid increasing in deep learning, more models tend to detect communities in either an end-to-end fashion or by formulating conventional models under deep learning frameworks. In this section, the six discussed tracks are the most popular ones and related studies are summarized in following paragraphs.

2.3.1 Modularity

As the most important metric to measure the fitness of a community partition, modularity reflects the superiority of how much the communities preserves in-community edges better than a random partition model. Mark Newman published a series of papers regarding to explain the modularity from various perspectives, such as edge connections and adjacency matrix transformation. Among these papers, [133] and [132] are two representative works to explain what is modularity and how to find out a proper community partition which maximizes the graph modularity efficiently.

Louvain method [133] so far is the most efficient method to find the optimal partition with largest modularity. Within the paper, modularity is interpreted as the measurement to compare the density of within-community edges with that of between-community edges. It is calculated as:

$$Q = \frac{1}{2|E|}\sum_{ij}(A_{ij} - \frac{k_i k_j}{2|E|})\delta(c_i, c_j) \tag{2.27}$$

where k_i is the degree of node i. $\delta(c_i, c_j) = 1$ if node i and j are in the same community, it equals to 0 otherwise.

The efficient Louvain method contains two phases and optimizes the modularity through an iterative approach. In the initialization step, all nodes are assigned to a single-node community. Then, for each node i, the paper considers to remove i from its current community and plug into one of the communities which its neighbor nodes belong to. It will be re-assigned to a community which has the largest modularity gain. The second phase will run iteratively until no further modularity gain achieved. [134] is a follow-up work of Louvain method to guarantee the generated communities are well-connected.

FPMQA model [135] is a parallel model to detect modularity based communities efficiently. The steps are similar as Louvain method. It also initializes a set of single-node communities, and merges communities which leads to the largest modularity gain in each step. The FPMQA model takes use of a mark array to store community states (busy or free) and merges communities based on their states in a parallel fashion.

In [132], it theoretically proves that the original modularity optimization problem can be rewritten as the eigenvalue and eigenvector calculation on a defined modularity matrix. In a two-community detection scenario, the modularity matrix can be written as:

$$Q = \frac{1}{4|E|} s^T B s \tag{2.28}$$

where s is a column vector in which $s_i = 1$ if node i is in community 1 or $s_i = -1$ if node i is in community 2. B is a symmetric matrix in which $B_{ij} = A_{ij} - \frac{k_i k_j}{2|E|}$. In the end, after matrix transformations, The final goal is to find a proper community assignment s to concentrate as much as positive eigenvectors of the matrix B.

A further approach, [136], solves modularity maximization from the nonnegative matrix factorization (NMF) perspective. It runs on modularity Laplacian matrix instead of the modularity matrix.

[137] extends the modularity to directed graphs for overlapping community detection. The modified modularity is calculated as:

$$Q = \frac{1}{|E|} \sum_{c \in C} \sum_{i,j \in V} [r_{ijc} A_{ij} - s_{ijc} \frac{k_{in} k_{out}}{|E|}] \tag{2.29}$$

where r_{ijc} is the weight of contribution of edge between node i and j to the modularity of community c. And s_{ijc} is the weight of contribution in the null model where communities are randomly assigned. The whole approach is optimized through a genetic process.

[138] introduces a local divisive heuristic to maximize graph modularity in undirected and unweighted graphs. It hierarchically divides the graph by using Kernighan-Lin heuristic, which proceeds a bipartition to reassign nodes from a community to the other. In each step, the bipartition reassignment which gains the largest modularity improvement will be selected. [139] also considers local graph connectivity and proposes a local modularity optimization to detect node communities.

[140] applies stacked auto-encode, a type of deep learning technique, to reconstruct the modularity matrix. Besides that, [141] introduces several other variants of modularity, including modularity density, fine-tuned modularity and fine-tuned modularity density. Modularity density avoids the resolution limitation problem of original modularity. And the two fine-tuned variants improves the measurement by splitting and merging the graph structures. Derived from modularity density, [142] proposes a new measurement named modularity intensity to indicate the cohesiveness of community partition, which is defined as:

$$M = \sum_{i=1}^{C} \frac{\alpha F(C_i, C_i) - \beta F(C_i, \bar{C_i})}{|C_i|} \tag{2.30}$$

where $|C_i|$ refers to the number of nodes in community C_i. $F(C_s, C_t) = \sum_{\forall i \in C_s, \forall j \in C_t} A_{ij} \cdot B_{ij}$. And $\bar{C_i} = C - C_i$. A is the adjacency matrix and B is the edge weight matrix, which is calculated as:

$$B_{ij} = \frac{\sum_{t=1}^{|V|} A_{it} \cdot A_{tj} + A_{ij}}{\sqrt{\sum_{t=1}^{|V|} A_{it} \cdot \sum_{t=1}^{|V|} A_{tj}}} \tag{2.31}$$

where $i, j \in V$ and $i \neq j$.

[143] shows a series of different tree and tree-like graphs, such as caley tree graph, z-ary graph and other clique or tree graphs. It theoretically proves that these types of graphs always obtain communities which have high modularity scores. And adding nontree-like components in graph will destroy this phenomenon.

To solve the resolution limit of modularity, [144] proposed a refined modularity metric by involving community degree factor, which compares the sum of average degree difference between the detected communities and randomly generated communities. The calculating formula is defined as:

$$Q = \sum_{k=1}^{K} \frac{\sum_{ij} (A_{ij} - \frac{k_i k_j}{2|E|}) S_{ik} S_{jk}}{\sum_{i=1}^{|V|} S_{ik}} \tag{2.32}$$

$S_{ik} = 1$ if node i in community k, otherwise it equals to 0. Compared with the original modularitym the only difference is that the normalized modularity divides the community size.

[145] is a very classic mathematical paper which proves the equivalence between modularity method and degree corrected stochastic block model under special parameter settings.

On the contrary to modularity, [146] proposes an anti-modularity metric to detect anti-communities in graphs. Anti-community is a particular type of node partitionwhere nodes with nor or few connections within the community and densely connected with nodes from extra-communities. The anti-community is defined as:

$$Q = \frac{1}{|V|} \sum_{c \in C} \sum_{i,j \in c} (\sum_{k=1}^{|V|} a_{ik} a_{kj} - \frac{k_i k_j}{|V|}) \tag{2.33}$$

where $\sum_{k=1}^{|V|} a_{ik} a_{kj}$ is the number of two-step paths between node i and j passing a third node k. The paper theoretically proves that the anti-modularity is fundamentally a principle component analysis method on the adjacent matrix. And in the end, the paper proposes a label propagation method to find a community partition with maximized anti-modularity. In detail, community labels are assigned to a set of seed nodes, which are propagated to other nodes through their connections. In the end, the nodes with same community labels are assigned to the same community.

2.3.2 Spectral Clustering

Spectral clustering is a type of approaches which leverages graph matrices (modularity matrix, Adjacency matrix or Laplacian matrix) to find out the top eigenvectors or other graph characteristics so as to learn low dimensional representations for nodes. By reducing the dimension of sparse graph matrices, it can preserve denser graph information and explore more cohesive node relationships. Spectral clustering has a huge relationship with stochastic block models. [147] proves that spherical k-median spectral clustering method can be extended the degree corrected stochastic block models. [12] is a survey paper which clarifies some graph terminologies and introduces several graph cut and spectral clustering methods. However, as it is published over a decade ago, the mentioned models are a bit dated and lack of detailed model explanation.

[148] is a theory paper shows that with a proper selection of parameters (i.e. how to choose the value of community membership matrix) in normalized cut via spectral method, it equals to the modularity based method and stochastic block model. [149] extends convolutional neural network (CNN) to general graph Laplacian matrix by adding a CNN operator on the eigenvectors of graph Laplacian. [29] shows that spectral algorithms on a defined nonbacktracking matrix performs better than on adjacency matrix or other first-order approximate matrices. The nonbacktracking matrix B is defined as:

$$B_{(u \to v),(w \to x)} = \begin{cases} 1, & v = w, u \neq x \\ 0, & otherwise \end{cases} \tag{2.34}$$

[150] introduces a spectral clustering method on Bethe Hessian matrix, which is also called as deformed Laplacian:

$$H(r) := (r^2 - 1)\mathbb{I} - rA + D \tag{2.35}$$

where $|r| > 1$ is a regularized term. Particularly, $|r| = \sqrt{c}$ is used in this paper where c is the average node degree in the graph. The eigenvectors of the matrix are calculated and those with negative eigenvalues in $H(\sqrt{c})$ or $H(-\sqrt{c})$ are selected. A standard k-means method is thereafter applied on these eigenvectors to generate node communities. Particularly, the negative eigenvalues of $H(\sqrt{c})$ shows the assortative communities, while those of $H(-\sqrt{c})$ represents the disassortative communities.

[57] proposes a spectral method for large-scale graphs by generating supernodes connected to the regular nodes. To reduce the graph size, it generates supernodes and connects them to regular nodes. supernodes are regarded as cluster indicators to detect regular node communities. In the end, the original graph turns to be a bipartite graph containing two types of nodes. To construct k supernodes, the paper firstly selects k seed nodes, and calculates all the rest nodes' shortest distance to these seed nodes in order to group them into k subsets as supernodes. The original graph can be converted to a bipartite graph to record the belonging of two node types where each

row refers to a supernode and each column refer to a regular node. After that, a SVD approach is applied on the bipartite graph adjacency matrix to learn low dimensional representations for all nodes. In the end, a k-means clustering methods finally helps to detect node communities.

[151] proposes a hierarchical and k-way spectral clustering method which decomposes top eigenvectors from constructed similar matrix W. W can be calculated as either a noisy hierarchical block matrix or a noisy k-Block Diagonal matrix. It proves the spectral clustering can tolerate the noises from similar matrix W and still achieves good community partition even W involves extra noise. [152] summarizes the mathematical theory and proofs behind spectral clustering and stochastic block models. It also points out the the significance of spectral clustering for graph visualization. [153] studies a spectral clustering method on graphs generated by Extended Planted Partition (EPP) model. EPP model generates graphs from a random graph with hidden community partition which affects edge generation probabilities. To facilitate the spectral clustering in such graphs, all graph nodes are randomly split into two sets P and Q. Nodes in Q are projected to the subspace generated by the bottom k eigenvectors of random walk based graph Laplacian of P, and further community detection can group nodes in Q to communities. The nodes in P are partitioned in a similar way.

[154] is a locally-biased spectral clustering method which involves extra constraints for community detection. It is formulated as:

$$\begin{aligned} & min\ x^T(D-A)x \\ & s.t. \quad \begin{cases} x^TDx = 1 \\ (x^TD\mathbb{I})^2 = 0 \\ (x^TDs)^2 > k \end{cases} \end{aligned} \tag{2.36}$$

where s is the constraint matrix, x is the node low dimensional representation aims to optimize. k is a threshold parameter. It offers a feasible solution which satisfies the local constraints.

[155] is a multiway spectral method maps modularity maximization to vector partitioning learning. Through a set of transformations, the paper finally aims to learn a representation r_i

for each node i which maximizes the modularity score Q in the graph where Q is defined as:

$$Q = \frac{1}{2|E|} \sum_{s=1}^{k} |\sum_{i \in s} r_i|^2 \tag{2.37}$$

s refers to each community and $i \in s$ means node i in community s.

[156] discusses the importance to choose regularization factors for community detection. It theoretically proves that the cluster discovery result is not fully depend on the minimum node degree. And regularization can better support to group low-degree nodes to well-structured communities. In this paper, Eigengap is defined as the gap of the k smallest eigenvalues to the rest of eigenvalues, which can be controlled by the regularization factors and in return will affect spectral clustering performance.

SCORE model [157] uses the coordinate-wise ratios between the largest eigenvector and the rest largest eigenvectors to construct a decomposed matrix for clustering, which significantly reduces the nuisance from degree heterogeneity problem. TSC model [158] is a spectral method on high-order graphs where graph information beyond edges implicitly connects nodes. Instead of finding out the pairwise relationship between nodes, it aims to detect node triplet relationships. Therefore, instead of constructing a 2-D matrix for matrix decomposition, it builds up a 3-D matrix to hold node transition probabilities.

SClump model [159] proposes a spectral clustering for heterogeneous graphs. It constructs a similarity matrix based on metapaths from [18]. [18] calculates pairwise node similarities for each metapath P_i, and sum them over to construct the metapath similarity matrix $\sum_i \lambda_i P_i$. Thereafter, it aims to learn a refined similarity matrix S that exhibits a clear clustering structure, the final objective function turns to be:

$$\min ||S - \sum_i \lambda_i P_i||_F^2 + \alpha ||S||_F^2 + \beta ||\lambda|| + 2\gamma \sum_i^k \sigma_i(L_S) \tag{2.38}$$

which is constraint with $\sum_{j=1}^n S_{ij} = 1, S_{ij} \geq 0$ and $\sum_i \lambda_i = 1, \lambda_i \geq 0$. $\sigma_i(L_S)$ is the i_{th} smallest eigenvalue of Laplacian graph L_S for matrix S.

[160] extends spectral clustering for signed graphs by constructing a family of Signed Power Mean Laplacians, which is defined by a transformation function of the normalized Laplacian graphs generated from both positive edges and negative edges. [161] is a scalable method utilizes graph random binning features (RB) and CoreCut model [162] introduces the relationship between regularized spectral clustering and graph conductance minimizing.

2.3.3 Stochastic Block Model

SBM is a major track of community detection method and there are so many papers regarding to this topic. As a type of approaches derived from statistical inference, most of papers in this track are theory papers aiming to prove their model efficiency under certain scenarios such lower & upper bound limitation. Hereby, I will broadly introduce the general goal in each paper instead detailed theoretically proofs. [163], as one of the most classic papers which introduces the SBM concept as a generative random graph model to reconstruct the original graph from communities. It assumes there exists community partition C and expected edge weight w_{rs} between node i in community r and node j in community s following Possion distribution. After a set of transformation functions, the final generative probability to maximize turns to be:

$$\log P(G|w, C) = \sum_{rs}(m_{rs} \log w_{rs} - n_r n_s w_{rs}) \tag{2.39}$$

where $m_{rs} = \sum_{ij} A_{ij} \delta g_i; r \delta g_j; s$ refers to the total number of edges between groups r and group s and n_r is the number of edges in group r. The paper also proves the equivalence between the standard SBM and modularity for undirected graphs. It further extends to a degree-corrected SBM to solve the limitation caused by node degree heterogeneity.

[9] is the latest survey paper published in 2017. It is a very detailed paper which introduces the general forms of stochastic block model. The main content discusses the thresholds during SBM phase transitions for exact recovery, weak recovery and partial recovery. It also extends to other related tracks of approaches such as graph-splitting, semi-definite programming and spectral

clustering. In the end, it points out several open questions such as possible extensions for semi-supervised or dynamic graphs.

[164] is a theory paper which discusses the graph reconstruction problem of sparse SBM with only two communities. The inter- and intra- edge connection probability are a/n and b/n where a, b are parameters and $n = |V|$ is the number of nodes. The paper proposes a belief propagation model to correctly assign node community labels under a general situation where $(a - b)^2 > C(a + b)$ where C is a constant. [165] also discusses the same binary SBM where there are only two communities in the graph. It converts the original community recovery problem to a transformed tree reconstruction problem. Particularly, it analyzes the density evolution of belief propagation on trees with Gaussian approximations.

[166] is an ensemble SBM model containing several existing SBM variants. It uses an entropy based log-likelihood function to infer community structure under two different scenarios including a soft constraint and a hard constraint. For a soft constraint, each imposed node degree refers to its average value among all SBM variants. While for a hard constraint, each imposed node degree should be exactly same among all variants.

[167] is an overlapping SBM which associates a latent vector for each node following multivariate Bernoulli distribution. The edge generative probability is calculated based on the latent factor of its start node and end node. [168] talks about the current limitation in partial or exact recovery of SBM and generalizes the discussion to overlapping communities.

[169] involves degree-corrected SBM to generate adjacency matrix which are used for spectral clustering. With proper adjustments in parameter selection, the regularized spectral clustering can achieve better performance for graphs where node degree significantly varies. It also points out choosing a parameter close to average degree can balance performance difference from several competing models. [170] is a theory paper which proposes a minimax rate to discuss the lower and upper bound for exact or partial recovery in SBM.

[171] checks the performance consistency for degree-corrected SBM. By comparing a set of different models, it proves that degree-corrected SBM can always obtain a stable model perfor-

mance. Modularity based methods need to have special parameter constraints in order to achieve a consistent result while likelihood-based methods do not. [172] is another paper checks the consistency of maximum-likelihood and variational estimators in SBM. It proves that in SBM variational estimators can be asymptotically equivalent to maximum-likelihood estimators to estimate the edge appearance probability between nodes. [173] focuses on the model selection in SBM as there are too many parameters to tune and general model-selection criteria can't properly work. The paper discusses the influence of different log-likelihood ratio distributions in both standard SBM and degree-corrected SBM. It further extends to sparse graphs to explore more graph scenarios. Moreover, it proposes a belief propagation based linear-time approximations for log-likelihoods, which proves to have relatively satisfied agreements.

[174] introduces a new labeled SBM task where there are labels appeared with a certain probability $p(l, i, j)$ between two nodes in community i and j. In the end, the paper proposes a spectral partition method under SBM to reconstruct the communities from the observation of these appeared labels.

[175] utilizes an optimized Markov chain Monte Carlo (MCMC) method to efficiently infer SBM in large graphs. It heuristically defines a node moving probability for community r to s as:

$$p(r \rightarrow s|t) = \frac{e_{ts} + \epsilon}{e_t + \epsilon B} \tag{2.40}$$

where t is the community label of a random node, e_t is its degree. ϵ is a tuning parameter and B is a relatively large constant.

Given a random labeled graph generated by standard SBM, [176] aims to infer its edge community distributions from node latent attributes. specifically, it proves that no model works well without observation if average node degree is below a certain threshold.

[21] proposes a SBM based edge community detection task and addresses the community size heterogeneity problem which is often ignored by previous works. In its definition, an edge $< i, j >$ is generated by two nodes i, j selected from community z. The node selection probabilities are θ_{iz}

and θ_{jz} and community size is w_z. In its generative model, a community z is firstly chosen with w_z nodes. And node i and j are selected in this community with the aforementioned probability to form an edge. The final expected number of edges between the two nodes in community z are calculated as:

$$\hat{A}^z_{ij} = w_z \theta_{iz} \theta_{jz} \tag{2.41}$$

And the final expectation of edge number between node i and j is the sum over all possible communities as $\hat{A}_{ij} = \sum_z \hat{A}^z_{ij}$.

[177] is a likelihood based SBM which can be extended to degree-corrected SBM with proper parameters. It calculates the asymptotic distribution under overfitting and uderfitting situation and proves its result performs stable when average node degree grows at a polylog rate. [178] is a goodness-of-fit test on SBM to offer a baseline result for comparisons with other competing models. [179] is an in-depth exploration particularly for degree-corrected SBM and proposes a polynomial time algorithm for asymptotic optimization in the degree-corrected SBM.

2.3.4 Deep Learning

In recent years, deep learning techniques are more and more involved in community detection tasks. Related approaches either leverage graph neural networks (GNN), graph convolutional networks (GCN) or other standard deep frameworks (i.e. autoencoder) to learn community embeddings or direct node-community distributions.

Autoencoder is a very intuitive approach to compress node original one-hot encoding to a latent low-dimensional space. By natural, the node latent vector can be regarded as its community distribution or be further clustered by other community detection models. [180] is the first work which uses autoencoder technique to learn node embeddings. It uses a four fully connected layers as the encoder to project original input to a latent space. Then a symmetric decoder is processed on the latent vector to reconstruct the original input. A locality-preserving constraint (from k-nearest neighbor nodes) and a group sparsity constraint (from same-community nodes generated from group lasso) are combined with the reconstruction error as the final objective to minimize.

In the end, k-means clustering method is applied to the latent node representation for community detection. Similarly [181] is a very classic method to learn node non-linear embedding via stacked autoencoder, and applies k-means to obtain node communities afterwards. [182] is another similar approach but with a shared weight in both encoder and decoder. And the final optimized weight is regarded as the node-community matrix. MGAE model [183] is a marginalized autoencoder which leverages both graph structure and content information into a unified GCN framework. The model considers both graph adjacency matrix and node content to learn latent node representations. And a spectral clustering is taken on top of the representations to detect the final node communities.

[184] jointly models three tasks to learn node embeddings, community embeddings and detect communities. The community embedding are generated from multivariate Gaussian distribution. And a Gaussian Mixture Model (GMM) helps to learn the generative probability of a node from a community. Meanwhile, the node embedding is optimized from a skip-gram model with negative sampling with the goal to generate nodes from its neighbor nodes. Similarly, [185] is also a multi-task generative model to jointly learn node embedding and detect communities. In its assumption, a node is represented as a mixture of communities, and a community is a multinomial distribution over all nodes. The generative probability of a neighbor node u of node v is calculated as:

$$p(u|v) = \sum_{c} p(u|c)p(c|v) \tag{2.42}$$

The probability is the sum over all conditions that node v generates a community c first, and community c thereafter generates node u. The whole generative process can be optimized under a deep framework using stochastic gradient decent optimization.

DLC model [186] proposes a single layer transformation as:

$$\min_{W,C} ||A - WDC||_F^2 + \lambda||C||_F^2 \tag{2.43}$$

where W is the linear transformation function, D is the dictionary in the linear coding, C is the code of the graph nodes. This linear transformation can be stacked into multi-layers to achieve a deep linear coding schema. The paper also theoretically proves its equivalence to the marginalized denoising autoencoding with spectral clustering.

ComE model [187] proposes an interesting task to learn community embeddings instead of node embeddings. Instead of representing communities as a vector and inspired by the Gaussian mixture model, ComE formulates that community embeddings follows multivariate Gaussian distribution with a tuple of a mean vector and covariance matrix. Therefore, the main output of this model is a set of community embedding (ψ_k, Σ_k) for each community $k \in \{1, .., K\}$. Node embeddings ϕ are learned as a prior knowledge via LINE method [188] to support community embedding. The whole process is optimized as a generative model which uses community embedding to generate node embeddings $p(v_i | c_i = k, \phi_i, \psi_k, \Sigma_k)$.

GEMSEC model [189] jointly learns node embeddings and clusters nodes into communities. It uses a negative sampling to maximize the generative probability of neighbor nodes given the current node. Meanwhile, it adds a community cost to the node embedding objective to learn community centers:

$$\text{argmin} \sum_{v \in V} \big[ln \Big(\sum_{u \in V} exp(f(v) \cdot f(u)) \Big) \big] + \gamma \sum_{v \in V} \min_{c \in C} ||f(v) - u_c||_2 \tag{2.44}$$

where $f(v)$ is the embedding of node v. $N_S(v)$ is a collection of nodes within a window size towards node v through random walks. The first term is the node embedding learning cost, and the second term is the community center distance cost.

DFuzzy model [63] is a three-step approach to learn fuzzy clusters. In the beginning, a pre-training step learns the initialized community centers through a personalized PageRank. And an autoencoder approach is taken with the PageRank result to learn the initialized node communities regarding as the center nodes. Second, modularity is used to redefine the partition result of initialized communities. Third, the community centers are updated based on the distance of the rest nodes in the community, which are calculated from the last layer of the model. The whole process will iteratively updated until convergence.

[140] learns node embedding from deep neural network, and extends its model to a semi-supervised community detection task by involving pairwise node contraints. The node embedding learning process is a standard stacked autoencoder approach to firstly project each node to a latent space and reconstruct it after that. To involve the pairwise node contraints, the objective function adds $\lambda Tr(H^T L H)$ to the reconstruction error in the stacked autoencoder. H is the learned node low dimensional vector, which is also regarded as node-community distribution. L is the Laplacian matrix of the node constraint matrix.

[190] is a very first work to really apply graph neural network (GNN) techniques in community detection tasks. It is a stacked model which contains multiple stacked components. In each layer of the component, it involves adjacency matrix in the non-linear transformation and the final output is node community labels. The whole approach can be optimized in an end-to-end fashion.

Cluster-GCN model [191] is an efficient model to leverage graph convolutional network (GCN) for community detection. It makes an innovation for the mini-batch SGD update by sample a subgraph to optimize the corresponding node embeddings. This strategy significantly reduces the computational cost in the orignal GCN framework without losing any graph information.

MRFasGCN model [76] solves a very complex task: using both graph convolutional network (GCN) and markov random field (MRF) for semi-supervised community detection in attribute graphs with semantic information. The model input contains adjacency matrix A, node attribute matrix X and node similarity matrix K calculated from A and X. The first two layers are GCN layers only involved with A and X and reLU activation function. In detail, the first layer is represented as $AXW^{(0)}$, the second layer is represented as $AX^{(0)}W^{(1)}$. And the third layer takes K into account as $KX^{(2)}W^{(2)}$. The last layer result is passed a MRF layer to learn the pairwise constraint between nodes.

For other works, [75] exploits the high-order graph convolutional networks and theoretically discusses the order influence to the model performance in attribute graph clustering. DMGC [192] is the latest work which detects communities in multi-graphs simultaneously via an attention module and minimum-entropy loss. CommunityGAN [193] jointly learns node embeddings and detects overlapping communities through a generative Adversarial net (GAN). The generator aims to generate motifs from nodes to approximate the real graph, while the discriminator aims to detect which is the fake motif generated from the generator or the ground-truth motif.

2.3.5 Matrix Factorization

A lot of matrices can be derived from graphs to reveal its structure, some of the most popular ones are degree matrix D, adjacency matrix A, Laplacian matrix$L = D - A$, normalized Laplacian matrix $D^{-1/2}LD^{-1/2}$, stochastic random walk matrix $Q = AD^{-1}$ and modularity matrix $M_{uv} = A_{uv} - d_u d_v / 2|E|$. Therefore, matrix factorization by nature can be directly utilized on these matrices to learn a hidden representation on nodes. Intuitively, if each dimension of the representation is regarded as a community, the node vectors can also indicate their affiliations in each community to solve the overlapping community problem. Inspired by this, nonnegative matrix factorization has been a popular track of methods for years.

[194] is the first work that utilizes nonnegative matrix factorization (NMF) for community detection. In its paper, it proposes three NMF based techniques including Symmetric NMF, Asymmetric NMF and Joint NMF. The Symmetric NMF is the simplest form of graph NMF, which assumes the graph is undirected so that adjacency matrix A is symmetric. The objective of Symmetric NMF turns to learn the low dimensional representation of nodes, which refers to the probability of all nodes belonging to communities:

$$\min_{X \geq 0} ||A - XX^T||_F^2 \tag{2.45}$$

In a directed graph, Asymmetric NMF is used to learn a node community membership matrix X and a diagonal matrix S which shows the connectivity within each community. Therefore, the objective function has a small change compared with Symmetric NMF method:

$$\min_{X,S \geq 0} ||A - XSX^T||_F^2 \tag{2.46}$$

The Joint NMF considers an even more complex scenario which considers a heterogeneous graph whose adjacency matrix A refers to the connections between the two types of nodes. U and D refer to the connections within each individual node type. Therefore, to learn a latent matrix X to uncover the belonging relationships of two type of nodes, it considers the all three aforementioned information:

$$\min_{X,\alpha,\beta\geq 0} ||A - X||_F^2 + \alpha||U - XX^T||_F^2 + \beta||D - X^TX||_F^2 \tag{2.47}$$

SymNMF model [195] comprehensively explains how to use the NMF to graph clustering. Derived from the standard NMF which decomposes on the adjacency matrix, it introduces another similar matrix $D^{-1/2}AD^{-1/2}$ and explains its equivalence to the Normalized Cut method. The paper also generalizes the similar matrix to a kernel function as $\phi(X)\phi(X)^T$. [196] is a follow-up work on the same SymNMF model but with more detailed supplementary. BIGCLAM [80], a previously mentioned overlapping community detection model, is also leverages nonnegative matrix factorization techniques.

[197] uses multi-step random walks to calculate node pairwise similarities. The original similarity matrix is the normalized Laplacian matrix $Q = D^{-1/2}AD^{-1/2}$ where D is the diagonal degree matrix. The j step node random walk similarity matrix is calculated as $(\alpha Q)^j$ where α is a decay factor. Summing over all possible j steps. the limitation is calculated as $\sum_j^{\infty}(\alpha Q)^j = (I - \alpha Q)^{-1}$. Therefore, the all step similarity matrix is used to replace the adjacency matrix in the objective function. The goal is to learn a low dimensional node-community distribution W by minimizing the reconstruction error to the similarity matrix:

$$\min_{W\geq 0} ||c^{-1}(I - \alpha Q)^{-1} - WW^T||_F^2 \tag{2.48}$$

where $c = \sum_{ij}[(I - \alpha Q)^{-1}]_{ij}$ is a normalizing factor.

[198] proposes a two-step matrix factorization approach. First, a singular value decomposition on the adjacency matrix is calculated as $A = U\Sigma V^T$. It selects the top ranked columns in the

three matrices to get a refined adjacency matrix N with more condensed information. A Bayesian Nonnegative Matrix Factorization (BNMF) [199] is thereafter applied on the matrix N by assuming each node in N follows a Poisson distribution. In detail, it calculates the posterior distribution of two smaller matrices W and H by maximizing the posterior criterion:

$$\min_{W,H,\beta\geq 0} p(W,H,\beta|N) \tag{2.49}$$

where β is a scale hyperparameter with Gamma distribution, and W, H are both with half-normal probability distributions.

BNMTF [200] is a bounded nonnegative matrix factorization approach which uses tri-factorization to decompose adjacency matrix A into three sub-matrices U, B, U^T. U is the node-community matrix where each row refers to a node distribution among all communities, which is bounded between 0 to 1. B is the connections between communities. The goal is to use the three matrices to regenerate a matrix $\hat{A} = UBU^T$ to let $A \approx \hat{A}$. The reconstruction error can be measured using either square loss l_{sq} or KL-divergence l_{kl}, which are calculated as :

$$\begin{aligned} l_{sq}(A,U,B) &= ||A - UBU^T||_F^2 \\ l_{kl}(A,U,B) &= \sum_{ij}(a_{ij}ln\frac{a_{ij}}{\hat{a_{ij}}} - a_{ij} + \hat{a_{ij}}) \end{aligned} \tag{2.50}$$

where a_{ij} is the related datapoint in A and $\hat{a_{ij}}$ is the estimated value of a_{ij}.

SBMF [201] not only detects overlapping communities via a binary matrix factorization, but also detects node outliers from unweighted graphs. Given its binary adjacency matrix A, $A_{ij} = 0$ if there is no edges between node i and j, otherwise $A_{ij} = 1$. The paper aims to find a binary community matrix U where $U_it = 1$ if node i in community t and 0 if not. If a node i belong to multiple communities, the sum of its vector will be larger than 1 ($\sum_t U_{it} > 1$). And $\sum_t U_{it} = 0$ if the node i is an outlier. In its model, it assumes there are a few outlier nodes which do not belong to any community, and the outlier nodes should be as few as possible. Therefore, from the basic matrix factorization objective function, it adds a term regarding the outlier node penalty and uses

L1 normalization in the matrix factorization objective:

$$\min_{U} ||A - UU^T||_1 + \sum_{i}[1 - \Theta(\sum_{j} U_{ij})] \tag{2.51}$$

where $\Theta(X)$ equals to 1 if $X > 0$ or 0 if $X \leq 0$.

[123] is a semi-supervised NMF approach which involves prior information (must-links l_{ml}) into the NMF objective function. The prior information constructs a constraint matrix M where

$$M_{ij} = \begin{cases} 1, & i = j \\ 0, & (v_i, v_j) \in l_{ml} \\ \epsilon, & others \end{cases} \tag{2.52}$$

The constraint matrix will be involved in the conventional objective function to learn the node-community matrix X:

$$\min_{X} ||A - XX^T||_F^2 + \frac{\lambda}{2}\sum_{ij} ||x_i - x_j||^2 M_{ij} \tag{2.53}$$

where x_i is the i_{th} row in node-community matrix X. [202] is a similar semi-supervised approach on unweighted, undirected graphs but with both must-links and cannot-links constraints.

Graph regularization is a typical strategy used on top of constructed graph matrices for enhancing the performance of matrix factorization models. DNMTF [203] is a novel graph dual regularization non-negative matrix tri-factorization model which considers the graph regularized terms from both structure based and feature based perspectives. A k-nearest neighborhood method helps to construct a data graph from graph topological structure, and a feature graph from node feature information. The objective function therefore includes three components:

$$\min_{U,S,V \geq 0} = ||A - USV^T||_F^2 + \lambda Tr(V^T L_V V) + \mu Tr(U^T L_U U) \tag{2.54}$$

where U, S, V are matrices need to be learned, L_v and L_U are related graph Laplacian matrices for data graph and feature graph. $Tr(\cdot)$ is the trace of related matrix. λ and μ are weights for the two regularized terms.

NMTF [204] utilizes three types of graph regularizations to capture user similarity, message similarity and user connections seamlessly in social networks. It contains three binary graph matrices ($M_{u-u}, M_{u-f}, M_{t-f}$), two similarity matrices (S_{u-u}, S_{t-t}) and one binary interaction matrix (R). The goal of this paper aims to learn a user binary cluster matrix U, message binary cluster matrix V and word binary cluster matrix W. By involving conventional NMF approach with three regularized terms, the overall objective function is defined as:

$$\begin{aligned}\min_{U,V,WH_1,H_2,H_3} ||M_{u-u} - UH_1U^T||_F^2 + ||M_{t-f} - VH_2W^T||_F^2 + ||M_{u-f} - UH_3W^T||_F^2 + \\ \alpha Tr(U^T L^u U) + \beta Tr(V^T L^t UV) + \gamma Tr(U^T L^\tau U)\end{aligned} \tag{2.55}$$

L^u, L^t, L^τ are the Laplacian matrix of S_{u-u}, S_{t-t}, R. And H_1, H_2, H_3 are three matrices to be optimized. To ensure a user/message can only belong to one cluster, the objective function should be constraint by $UU^T = I$ and $VV^T = I$.

MHGNMF model [205] extends the graph regularizations to hypergraphs by considering high-order node information to enhance model performance. RGNMF [206] considers an extra error matrix S in the conventional NMF approach:

$$\min_{U,V,S} ||A - UV^T - S||_F^2 + \gamma||S||_1 + \mu \sum_{ii'} W_{ii'}^U ||U_i - U_{i'}||_2 + \lambda W_{jj'}^V ||V_j - V_{j'}||_2 \tag{2.56}$$

The last two terms are related graph regularized terms.

DANMF model [207] is a deep autoencode-like method to learn node communities via an encoder-decoder component. It is a very straightforward method which uses multiple layers to transform the original adjacency matrix to a low dimensional community space (decoder component). Then it uses the node-community matrix to reconstruct the original adjacency matrix (encoder component). The overall loss it the combination of two component errors with a regularization term. M-NMF model [208] incorporates a NMF based node representation learning model and a modularity based community model together to optimize them jointly.

2.3.6 Flow-Based

Flow-based models assumes either random walks or information propagation in the graph. Through these flow-like process, the energy or information of each node will be propagated to its neighbor nodes. When this process becomes stable, the nodes contains similar type or amount information will be grouped into the same community. By nature, flow-based models are all Markov chains, as the next step node will be fully dependent on the current step node. This type of approaches are quite scattered, random walk models, local search models, Potts model, and heat kernel based models all fall into this category.

Map Equation [209] is a metric to quantify how well a community can compress graph information. Derived from this metric, there are several variants are proposed to serve other more complex scenarios such as hierarchical community detection [210], dynamic community detection [211] and sparse Markov chain to solve vast parameters problem in high-order Markov chain [212]. Particularly, [211] shows second-order Markov dynamics in the random walks can lead to significant influence for community detection, node ranking and information propagation.

In detail, [209] uses a probability of random walks as an proxy of information flow in the graphs by introducing the Map Equation metric. In order to describe a random walk path, it utilizes huffman codes to encode nodes by assigning shorter codewords with hub nodes and longer codewords to rare nodes. And a path can be described as the concatenation on the codewords of all nodes appeared in the path. The goal in this approach aims to find an optimized community partition C which group all nodes into k communities, which has the minimum average description length of all the paths $L(M)$. The metric is defined as:

$$L(M) = q_{\curvearrowright} H(\mathcal{L}) + \sum_{i=1}^{k} p^{i}_{\circlearrowright} H(\mathcal{P}^{i}) \tag{2.57}$$

This metric contains two parts: first is the entropy $H(\mathcal{L})$ of the random walks between communities, and second is the entropy $H(\mathcal{P}^i)$ of random walks within each community (where exiting current community also is considered a step in random walk). To minimize the Map Equation metric, a deterministic greedy search algorithm is proposed and refined via a simulated annealing process.

[210] extends the original map equation to a hierarchical version. For a hierarchical map M with k communities, each community i contains a submap M^i and m^i sub-communities, the hierarchical map equation is calculated in a nested way:

$$L(M) = q_{\curvearrowright} H(\mathcal{L}) + \sum_{i=1}^{k} L(M^i) \tag{2.58}$$

UEOC model [213] uses Markov random walks with constraints strategy to detect overlapping communities. It contains four steps: first, select the node with maximum degree and non-community membership; second, apply random walks on the selected nodes with a constraint number of steps. Calculate the probability of each node being the end node and rank them; third, select nodes with conductance score larger than a threshold to be in the same community of target node; fourth, if there are still nodes without being assigned to at least one community, repeat the step 1 until no such nodes remained.

[214] introduces a heat kernel based diffusion model to detect communities in a local and deterministic manner. The overall goal of this approach is to approximate a heat kernel h as a diffusion function in the graph:

$$h = e^{-t} \left(\sum_{k=0}^{\infty} \frac{t^k}{k!} (P)^k \right) s \tag{2.59}$$

where $P = AD^{-1}$ is the random walk transition matrix and D is degree matrix. t is a coefficient term. And s is an initialized seed vector which sums to one. In the end, the subgraphs with low conductance score will be regarded as single communities.

[215] studies a particular Topologically Biased Random Walk (TBRW) on graphs for commu-

nity detection. A standard transition matrix T is calculated as:

$$T_{Ij} = \frac{W_{ij}}{\sum_l W_{lj}} \tag{2.60}$$

W_{ij} is the edge weight between two nodes i and j. In a biased random walk, the edge weight can be defined as $W_{ij} = A_{ij}e^{\beta x_i}$. x_i can be any defined factors such as related node degree. After the transition matrix is calculated, spectral clustering can be leveraged to select top eigenvectors as node community distributions.

[216] proposes two simulated annealing algorithms,SADI (dissimilarity-index-based) and SADD (diffusion-distance-based), to generate communities with maiximized modularity under a k-means framework. A dissimilarity index measures the proximate extent between each pair of nodes. And diffusion distance metric calculates the distance difference for each pair of nodes to the rest nodes in the graph. Each community centroid is calculated given a certain community partition and related metrics. After that, a simulated annealing k-means function is applied to find the best partition with largest modularity based energy score. Nodes will be merged or separated in an recurrent way according to the gain or loss of the graph modularity.

[217] uses Potts model via a Markov process which assumes nodes as spins and simulates spin dynamic value changes trough spin-spin correlations. The detected dynamics are used to detect node hierarchical communities. [218] ranks nodes and detects their communities via PageRank algorithm with teleportation. Instead of standard teleportation to nodes (the next step of a random walk has a small chance to a random-selected node instead of direct-connected nodes), this paper proves the teleportation to edges can achieve a more robust community partition result.

[219] is a fuzzy overlapping community detection method based on distance matrix calculated via local random walks. A t step local random walk (LRW) index between two nodes i and j is defined as:

$$s_{ij}^{LRW}(t) = \frac{k_i}{2|E|} \cdot \pi_{ij}(t) + \frac{k_i}{2|E|} \cdot \pi_{ji}(t) \tag{2.61}$$

where k_i is the node degree of i, $\pi_{ij}(t)$ is the probability of a t step random walk with start node i

and end node j. Derived from local random walk index, the t step superposed random walk (SRW) index between i and j is calculated as $s_{ij}^{SRW}(t) = \sum_{l=1}^{t} s_{ij}^{LRW}(t)$. Subsequently, each datapoint d_{ij} of the distance matrix D is calculated as:

$$d_{ij} = \begin{cases} 1 - s_{ij}^{SRW}(t), & i \neq j \\ 0, & i = j \end{cases} \tag{2.62}$$

After that, a standard multidimensional scaling (MDS) method is applied on D to project each node into a low dimensional space. An existing fuzzy-means (FCM) method helps to learn each node fuzzy communities from the projected space.

[220] discovers that three to four steps closed walks in the graph can reveal community structures veiledly. A closed walk is defined as a random walk with same start and end node. Such as a walk '1-2-3-1' is a closed walk while '1-2-3' is not. It measures an edge importance score by involving three step and four step closed walks, which is defined as:

$$s_{ij} = \frac{z_{ij}^{(3)} + 1}{min(k_i - 1, k_j - 1)} + \frac{z_{ij}^{(4)} + 1}{min(k_i - 1, k_j - 1)} \tag{2.63}$$

where $z_{ij}^{(k)}$ refers to the number of k step closed walks that the edge e_{ij} participates in. Edges are iteratively removed from the graph according to their importance score, and the remained disjoint graph components are naturally regarded as communities.

For other works, [221] is a flow based local partition method which is a refined model and theoretically proves its efficiency and effectiveness. [222] explains second-order Markov process in graph random walks can better reveal community structure in dynamic graphs. NetMRF [223] is a Markov random field model which infers node communities by belief propagation. [224] proposes a class of non-linear diffusion function between nodes, and compares their performance with non-linear transformation functions in neural networks.

2.3.7 Summary

In this section, I explore major tracks of community detection solutions and introduce the most representative works in the recent decade. In fact, these tracks are not fully independent with each other. For example, spectral clustering and stochastic block models share similar mathematical forms. And modularity methods start to be more inveloved in deep learning methods . A graph can be viewed as either a matrix to record nodes pairwise relationship or a flow where information propagates through edges. All tracks of methods are developed from these two graph understandings. There are some other tracks of approaches as well, which are either similar to existing tracks (graph cut methods is similar to spectral clustering), or classic models waiting for more exploration (information theory models).

2.4 Applications

In this section, two types of applications are discussed from both interdisciplinary and technical support perspectives. For interdisciplinary supports, community detection can be applied in other domains to support better domain knowledge exploration. For technical supports, various public dataset and open-source toolkits are introduced for model comparison and evaluation. The following subsections will introduce each type of supports in detail.

2.4.1 Interdisciplinary Supports

Social media is a major relevant domain towards community detection as user connections can naturally construct social networks. Besides that, other domains such as biology, physics and neural science also involve community detection a lot to explore their object hidden connections. In the following paragraphs, social media researches are separately introduced at first, and the rest relevant researches are demonstrated together afterwards.

Social Media

[225] introduces and compares a set of community detection models and provides five strategies for how to apply these models to support large scale social media analysis, including sampling techniques, local graph processing, iterative schemes, multi-level approaches and parallelization. These techniques are applied to five types of social media applications including topic detection, tag disambiguation, user profiling construction, photo clustering and event detection.

In collaborative tagging (a.k.a. folksonomy) systems, tripartite graphs can be constructed from users, images and tags given user behaviors in the systems. By defining random walk probability values between different types of nodes, [226] detects latent user communities in folksonomy graphs through a proposed Approximate Prototype Clustering (APC) method which is a similar process as K-means method. The tags of neighboring community users are ranked and selected to enrich a user's profile.

Due to the fact that some users create multiple sockpuppets to deceive other users or manipulate topics in online communities, [227] studies sockpuppetry to show the behavior difference between sockpuppets and ordinary users and thereafter detect sockpuppets to maintain the correct order of online communities. By analyzing 9 online communities, the study demonstrates particlar sockpuppets writing patterns (more singular first-person pronouns), write shorter sentences, and swear more) and behaving patterns (participate more controversial topics and more interact with other sockpuppets). In the end it builds up the taxonomy to differentiate and kick out sockpuppets for improving the quality of online discussion.

[228] studies individual user lifecycle evolving trend and linguistic change in online communities. In the analysis of its proposed framework, during the early stage (about one third of eventual lifespan) of individual users, they will tightly and increasing follow and be affiliated with current communities. However, after reaching to the peak point, a gap between the users' language and community forms increases until finally they abandon the site. The main reason is because of linguistic changes. Language norms used in a community is evolving over time, and it would be harder and harder for senior users to accept this change. Once users feel themselves not able to

cease the linguistic change, they will stay away from the community and eventually leave out. Instead of analyzing individual users in online communities, [229] focuses on the lifecycle of online community itself. It finds out two types of community growth including diffusion growth, where new members come in because of connections with existing members, and non-diffusion growth where new members come in without previous connections to the communities. It also builds up a model to predict community longivity and eventual size using graph structural features and past growth experiences.

Users in social media such as Facebook and Twitter needs to identify their social circles either manually, or in a naive fashion by shared attributes, which is not enough to satisfy users need and lack of accuracy. [230] proposes an ego network model to define user social circles based on different aspects (i.e. family members or schoolmates) at first. Then it assigns user followings/followers into different social circles based on their profiles. It allows overlapping communities where a user can belong to multiple types social circles to current user.

Many of existing works jointly model textual content and user activities in social networks to enhance the appropriateness of detected communities. [231] presents a step-by-step approach in YouTube graphs by clustering videos into named clusters having associated tags and descriptions. It prepossess the YouTube graph and selects seed nodes using a greedy method. After that, in order to enable community detection on millions of videos efficiently, it takes advantages of MapReduce to cluster videos in a parallel fashion. In the end, a post-processing approach is applied to refine and merge clusters by maximizing text coherence and minimizing community overlap. Having a similar goal to detect topic oriented communities, [232] extracts social objects from emails and blogs. Based on their content, social objects are clustered into different topics. Users whose behaviors are involved in the same topic are grouped into same communities. [233] also aims to detect topically meaningful communities by considering both graph topological structure and social content in a united way. Inspired by topic modeling, it proposes a generative Bayesian model which is named as Topic User Community Model (TUCM). In the model, it assumes community membership is dependent on both topic interests from posts and graph structure. The whole learning process is guided by a Markov Chain Monte Carlo (MCMC) sampling strategy. In the end, a person can belong to multiple communities and each community can contain multiple topics. For the same purpose, [234] proposes an improved version of probabilistic model to jointly detect user communities and content topics by leveraging both their social connections and shared content in Twitter. In its model, community is treated as an affiliation probability distribution on users. User connections as well as associated communities topics are generated from community structure by utilizing Gibbs Sampling. Instead of using topic modeling, [235] addresses Non-negative matrix factorization method which leverages social connectivity and content information for community detection. Besides user connectivity, It considers three types of content information including word, hashtag and domain as auxiliary terms to regularize the matrix factorization process. Its empirical experiments show that word usage is the strongest indicator of user potential community affiliation among all three types of content information.

Mobile devices have more and more become an essential part of our daily life with the proliferation of wireless technologies. Typically, a mobile social network (MSN) is constructed by user call logs. [236] introduces a Community-based Greedy algorithm (CGA) model which contains two major components including a community detection model to take care of information diffusion, and a dynamic programming model to select top K most influential users given the community partition. [237] studies on evolutionary MSN over time, reveals similar circadian and weekly patterns happened in both individual user level and community level.

[238] studies the graphs of Facebook "friendships" at five U.S. universities. It investigates the community structure of each single university graph and employs visualization and quantitative analysis to measure the correlation between user communities across universities. After examining the community impact of a set of self-identified user characteristics such as residence, class year, major, and high school in all university graphs, the study concludes that student relationships are organized and dominated by multiple key factors instead of a single one.

Miscellaneous Domains

[239] demonstrates how modularity method detects node communities within brain graphs to reveal human neural systems functioning on the main healthy human cognition. In this study, the defined nodes in a brain graph can be flexible but need to implicate network dynamics, such as brain regions or neurons. A d edges can reflect structural connections across spatial scales, such as bundles of axonal fibers between regions or synapses between neurons. Similarly, [240] also presents how community detection methods employed as neuroscience applications to identify the functional brain modules from multichannel and multiple subject neuroimaging data. It proposes a refined model based on modularity to detect communities in effective brain connectivity graphs. The paper quantifies brain connectivity with a defined directed information (DI) metric. It thereafter extends the Louvain method in a group of brain graphs to detect the most involved functional electrode modules in cognitive control.

Community detection techniques also contribute a lot in the biology domain to discover the hidden modules and potential bindings between proteins. [241] proposes a EGCPI model to identify protein complexes in the detected clusters from protein-protein interaction graphs. Based on the public Gene Ontology (GO) database, this paper constructs a protein attribute graph. With an evolutionary strategy to maximize the Independence of Cluster (IoC) fitness function, protein clusters are achieved through a genetic framework. In the end, a breadth first search (BFS) approach is applied to select sub-graphs that consist of similar proteins in each cluster based on their degree of attribute homogeneity. Having a similar propose, ClusterONE [242] is step-by-step approach to detect overlapping protein complexes from protein-protein interaction graphs. Specifically, it firstly utilizes a greedy approach to group proteins with high cohesiveness. Secondly, a merging process is taken on pairwise raw protein clusters to merge those clusters with high predefined overlap score. At last, it leaves out those small clusters with trivial influence in graphs. [243] also reaches a conclusion that community structure in protein interaction graphs can benefit biological findings by probing different scales/resolutions in the graphs.

[244] proposes a generative model to detect communities in the DBLP dataset, which is the

largest computer science bibliography database. By detecting communities in a constructed heterogeneous bibliographical graph involving author, paper, conference and particular word/term nodes, it calculates the continue, merge and split rates of words/terms as well as authors in the graphs over time. It also shows how evolutionary communities are appeared and disappeared as well. Another work, OverCite [245], also applies community detection in bibiliometrics, which aims to detect overlapping communities of authors, papers and venues simultaneously through the graphs constructed by these types of nodes. After detecting all node communities, the study builds up a recommendation system to use the overlapping communities as recommendation results to users.

[246] proposes a CENFLD model to deal with community detection in business/enterprise graphs. A business graph involves user nodes to represent producers, suppliers, and customers with side textual information such as recruiting messages or advertisements. To deal with business graphs' intrinsic nature of diversity, inconsistency, Implicitly and richness, the paper proposes a co-clustering factorization based approach to detect user groups. Specifically, it employs a non-negative matrix factorization method to factorize the graph topological and textual information in individual forms (including node-feature content matrix factorization, network topology structure matrix factorization and feature-feature correlation matrix factorization). After that, it proposes a consensus principle to optimize these forms jointly.

Besides aforementioned applications, community detection can also be applied in the chemical domain to discover force-chain clusters in granular materials [247], music domain to extract musical rhythmic pattern [248], and question-answer system [249] to improve Q&A matching accuracy.

2.4.2 Technical Supports

After understanding how those state-of-the-art models work, a subsequent task for researchers is how to employ these models on various graphs. As most of models are too complex to be implement by individual researcher within a short period of time, open-source softwares and packages are required with an urgent need. Similarly, benchmark graphs are also an essential part of model evaluation as they enable to testify different models under a fair condition. In the following paragraphs, I will list and detailedly introduce several publicly available graph repositories, widely used softwares and programming toolkits. All of them significantly ease and contribute to the evaluation process by offering an environment to compare model performances under different graph scenarios.

Datasets

SNAP [250] is one of the most famous graph repository which contains hundreds of graph datasets collected by Stanford University[1]. Within the graph repository, there are graphs with different semantic meanings such as social networks, citation graphs, communication graphs and web graphs. Regarding to the graph types, it includes signed graphs, temporal graphs and attribute graphs, etc. In total, the repository contains over 70 graph datasets which size range from thousands nodes to millions nodes. Many community detection papers published by Stanford University reply on this repository.

KONECT [251] is an open project held by the University of Koblenz–Landau, which particularly collects large graph datasets for academic research[2]. Up until now, there are 261 graphs with various types including (un)directed, (un)weighted, and signed graphs, etc. These graphs are also associated with different semantics such as social networks, citation graphs and communication graphs.

LAW graph repository[3] is owned and managed by the University of Milan, which particularly focuses on big graph storage. In its repository, there are around 80 big graphs compressed via WebGraph, a graph compression framework, to enable their quick downloading. There are various types of graphs such as web graphs, social graphs in the repository, each of which contains up to 100 million nodes and over 3 billion edges. Even though the repository official website is still

[1]http://snap.stanford.edu/data/
[2]http://konect.uni-koblenz.de/networks/
[3]http://law.di.unimi.it/datasets.php

maintained by the host, after the year 2012, there is no major updates in either graph datasets or relevant published papers.

NR [252] is an interactive graph repository which enables graph downloading and interactive web analysis through their web-based platform[1]. Currently, NR contains over 6000 graphs from 19 general domains, i.e. social netoworks, biological graphs. It covers a wide range of graph types such as bipartite graphs and temporal graphs as well. With the help of their interactive web interface, researchers can easily discover these graphs without too much effort. The lightweight platform also allows researchers to upload and share their own graphs. In this repository, the size of graph nodes is from hundreds to over 10 million. And the size of graph edges is from hundreds to over 100 million.

There is a repository of Benchmark Graph Datasets in Github[2], in which there are 31 medium-size graph datasets (each graph contains up to 10 thousand nodes). The types of graph include biological graphs, citation graphs, social networks and brain graphs. According to the repository, a series of models are assessed by graph classification and community detection evaluation tasks using these graph datasets.

Softwares

Pajek[3] is a public network analysis toolkit particularly for analysis and visualization on very large graphs. It is a well-maintained nonprofit project and the latest version is release in September, 2019. This tool offers hands-on manuals in all major languages including English, Chinese and Spanish. Originally, this software is deigned to run on Windows system. However, in recent years, several extensions are built up to enable running Pajek in both Mac and Linux system as well. Pajek is one of the best known graph analysis toolkit. The eligible graph format created by Pajek even becomes a standard format of graph data. Besides visualizing graphs into a 2-D plot, it can also calculate several major graph metrics such as triangle numbers and graph modularity score.

[1]http://networkrepository.com/networks.php
[2]https://github.com/shiruipan/graph_datasets
[3]http://mrvar.fdv.uni-lj.si/pajek/

Several basic community detection, such as Louvian method, are also embedded in Pajek to reveal high order graph structure.

NetMiner[1] is a commercial software for exploratory analysis and visualization on graphs. It is owned by a Korean company named CYRAM. The software is similar to Pajek software but has more interactive functions. As one of its advantages, It supports large network analysis on graphs with thousands of nodes, which can satisfy most of needs in the social science domain. Moreover, five community detection methods, including Modularity method and label propagation method, are implemented in the software to support more intricate graph analysis.

CFinder[2] is a free graphic tool to find graph communities and visualize the generated node cliques in 2-D plots. It is developed based on Java Spring framework and has a similar layout to other previously mentioned softwares. It requires to install Java Runtime Environment (JRE) as a prerequisite. CFinder is able to run under all main platforms (i.e. Windowns, Mac and Linux) and can be easily installed by researchers with no technical background. The latest version is released in the year 2014 and the latest paper is published in 2016. To summary, it is a possible software option for researchers to explore and visualize graph communities but the back-end support is a little bit out of date.

Gephi[3] is a leading visualization and exploration tool for social networks. The goal of Gephi is to make better analysis to find patterns to uncover graph structures and visualize the result in an intuitive way, which makes it easier for users to understand. In its interface, there is a visualization window along with a bunch of functional buttons. By clicking on the buttons or change the values of the related boxes, it can directly change the visualization layout. Even though Gephi offers a Python library as well, its software is still more popular and better accepted by users. One advantage of the software is that dynamic networks can be also conveniently explored. Another advanced feature is that Gephi is able to render 3-D plots in the visualization window. The recent major update is in 2017, which is able to support all types of platforms including Windows, Mac

[1]http://www.netminer.com/main/main-read.do
[2]http://www.cfinder.org/
[3]https://gephi.org/

and Linux.

UCINET[1] is a software funded by a start-up company named Analytic Technologies. It is particularly designed for Windows system and needs to be purchased after 90 days free trial. This software is pretty active and the latest version is release in March 2020. Although the official website claims the software can handle graphs with up to 30 thousand nodes. However, empirical experiences show that its process will run slow when the graph size is over five thousand nodes.

GUESS[2] is an exploratory data analysis and visualization tool for graphs. Claimed in its official website, the software contains a script language called Gython to handle large graph visualization via Java Applet. There are also several community detection methods embedded in the software, which can help to visualize graph hidden structure. However, as there is no major update since 2007, researchers have abandoned this software. Even though, it still offer some insights for graph analysis. Unlike other softwares only need users to click buttons in the interactive interface, GUESS requires users to type commands instead for generating the graph plots. This feature raises the bar to learn this software.

ORA-LITE[3] is a tookit for dynamic graph assessment, which is developed by the Carnegie Mellon University . It is a software mainly used for dynamic networks, which means the graphs they analyze change over time. As it claims in the official website, this software can not only find out the community structure of dynamic graphs but also other graph metrics. Although the time complexity is usually high in dynamic network analysis, this software is able to handle graphs with millions nodes. Another advantage of this software is that it offers multi-language tutorials and even holds Google groups for users to communicate. The latest version of this software is published in January 2020. But it only support to run on Windows system.

Cytoscape[4] is an open-source platform originally designed for biological researches. Through the development in recent years, it currently turns to be a general platform for all types of graph analysis and visualization. Cytoscape offers a Javascript API to allow rendering graph plots in web

[1]https://sites.google.com/site/ucinetsoftware/home
[2]http://graphexploration.cond.org/
[3]http://www.casos.cs.cmu.edu/projects/ora/
[4]http://www.cytoscape.org/

applications and a Java API to allow connecting to remote servers. Inside its software, there are a set of basic features installed by default. Beyond that, users have a choice to enable advanced features by installing plugins by themselves. The latest version of this software is released in June 2019. And based on the announcement from its official website, the number of software daily downloads is huge and its discussion forum is pretty active until now.

MuxViz[1] is a framework for the multilayer graph analysis and visualization. It allows interactive visualization and exploration for graphs with multi-type relationships and attributes, which is the most outstanding feature of this software. MuxViz is developed based on R and GNU Octave, which means it requires to install R in advance and can only deal with at most middle-size graphs. As a open-source software, it can run on Windows, Linux and Mac OS X. However, its latest release is in 2015, which is a bit out of date.

Visone[2] is a free toolkit for analyzing and visualizing social networks, which is a long term project managed by the University of Konstanz. It can run either from terminal or through its interactive interface. The software supports all types of platforms including Windows, Mac and Linux. As It is originally written in Java, the current version released in 2019 requires at least Java 8 to be installed in advance. Many community detection methods are embedded in it such as Modularity and spectral methods. Another feature is that the rendered plots can be directly exported and saved to local disk.

Programming Toolkits

As one of the most popular programming language in statistics, R contains many packages which are particularly designed for graph analysis and community detection. Wrapped as a high level programming language, R is not able to handle large scale graphs as other languages such as Java and C++. However thanks to its simplified design, it is able to support more complex models than other languages. In the following paragraphs, I will briefly cover three widely used packages including modMax, networktools and RANN.

[1]http://muxviz.net/index.php
[2]https://visone.info/

modMax[1] is a R packages which contains a lot of Modularity based methods. As one of the largest methodological track in community detection, Modularity has a lot of variants and optimization solutions. In modMax, many variants are covered such as Modularity optimization using fast greedy, simulated annealing, spectral and genetic methods.

networktools[2] is another newly developed R package particularly for graph analysis. Its functions include not only basic community detection algorithms but also have visualization functions.

RANN[3] provides a set of fast k nearest neighbor search models. It is a R wrapper of ANN library, which is originally written in C++.

igraph[4] is one of the most widely used packages for community detection. It is originally written in C++ but offers R, Python and Mathematica wrappers. igraph package is frequently updated and the recent release is released in March 2020. There are a bunch of widely used community detection models implemented in the package such as Modularity, Walktrap, Label Propagation and Infomap method.

NetworkX[5] is a fundamental Python package which defines the basic data structures used for graph analysis. Most of Python packages related to graph analysis are inherited from NetworkX. Within the package, it also contains several components such as centrality, clique, clustering in which several functions are related to community detection models or graph evaluation metrics.

SNAP[6] is a general graph mining library. It is written in C++ and easily scales to massive networks with hundreds of millions of nodes. The package has both C++ version and a Python wrapper. So far, it is the quickest package for community detection in large scale graphs. Another advantage of this package is that its embedded models keep updating frequently. It contains tens of state-of-the-art models for overlapping community, dynamic graph community, and Modularity based community detection.

[1]https://cran.r-project.org/web/packages/modMax/modMax.pdf
[2]https://cran.r-project.org/web/packages/networktools/networktools.pdf
[3]https://github.com/jefferis/RANN
[4]https://igraph.org/
[5]https://networkx.github.io/
[6]http://snap.stanford.edu/index.html

CDlib[1] is a newly announced Python library for complex network analysis. The recent version is released in February 2020. This software is extended from NetworkX but offers more community detection methods and evaluation metrics. Overall, It contains over 40 community detection methods about node clustering, edge clustering and overlapping community detection. Besides, it also offers functions to calculate more than 20 widely used evaluation metrics.

2.4.3 Summary

In this section, I firstly introduce how community detection can support researches in other domains such as citation recommendation, biology exploration and social media analysis. After that, a bunch of public resources, such as datasets and toolkits, are introduced to support community detection model evaluation. Softwares are easier to learn but programming toolkits have more flexibility to deal with complicated scenarios. Here are some tips and hints about how to choose toolkit if needed:

First, toolkit selection is purpose oriented. Researchers need to clarify their research tasks, and thereafter to select the proper toolkit. For example, if researchers need to analyse dynamic graphs, ORA-LITE is definitely the first choice to consider.

Second, identify the preference of flexibility or simplicity. If researchers are lack of technical background or don't want to spend too much time for learning, then softwares should be a better choice than programming toolkits. While if researchers want more customized functions, programming toolkits will be a better choice.

Third, price matters as well. Some paid softwares may contain more features than free ones. Researchers have to make sure whether it is worthy to invest on these commercial softwares such as UCINET.

Fourth, keep updated. There are many toolkit came out each year. Researchers need to keep up with the latest toolkit and explore the possibility to apply it into personal researches. Usually toolkits released by high-tech companies and top tier universities are associated with enriched

[1]https://cdlib.readthedocs.io/

tutorials. They are easier for new users to start learning and more reliable to use compared with other open source packages shared via Github repository.

Overall, researchers need to consider all above mentioned tips before they choose datasets and softwares to use in their academic researches. The appropriate selection may significantly alleviate their workloads in model performance analysis.

2.5 Evaluation

Once the community partition result is retrieved by taking a certain model, a subsequent approach is to evaluate model performance by running comparisons among different models. Therefore, how to carry out comparative analysis and what metrics should be reported as benchmarks are two imperative factors to be considered. In the following sections, I will address these two questions separately by introducing several highly cited papers about comparative analysis and a number of best known metrics widely used in industry and academia.

2.5.1 Comparative Studies

Comparative studies often run a number of methods on different datasets to explore what factors to cause the performance differences. This type of research aims to offer model suggestions according to graph structure. As a follow-up work of [253], [254] compares 13 community scoring functions and employs them on 230 graph datasets belonging to various topics (i.e. social graphs, collaboration graphs, and purchasing graphs) and with different sizes (from hundreds of thousand to hundreds of millions of nodes and edges in graphs). The 13 scoring functions lie in four categories including internal connectivity, external connectivity, the combination of internal & external connectivity and network model (modularity). To assess these scoring functions' community adequacy, four metrics are thereafter to considered, including separability, density, cohesiveness and clustering coefficient , which are often used to represent structural-level community coherence and compactness (details will be introduced in the next section). There are significant performance differences among all scoring functions: conductance (it measures the ratio of edges linking outside of the community) and triad participation ratio (the ratio of nodes in the same cluster forms a triad) achieves the best performance, while modularity is with poor performance.

[255] argues that current community detection methods totally ignore the topological nature of the communities as two methods may have similar evaluation performance but totally different community distributions. It claims that adding community topological metrics such as community-wise density, average distance, internal transitivity, and hub dominance may give more supportive understandings about the community partition result. Moreover, by testing on several real-world graph datasets as well as synthetic graphs using different types of approaches (e.g., modularity-based and diffusion-based approaches), it also empirically shows that there is no direct correlation between the model performance metrics (e.g., rand index and normalized mutual information) and community topological metrics.

[256] questions the appropriateness to simply use the node metadata information (e.g., node category or membership) as the criteria to build up the ground truth. By running 11 different community detection models on 16 different graph datasets, it claims that there is a trivial correlation between the communities constructed by graph metadata and model-detected communities. Thus, it concludes that metadata communities are not able to fully infer graph topological structure as they are separate views for the graphs. Instead, graph metadata can be taken as either auxiliary information to enhance community partition performance or another perspective to interpret graphs.

By employing 8 different models on Lancichinetti-Fortunato-Radicchi (LFR) benchmark graph [257], [258] summarizes how well these models can handle large-scale graph and community size recovery problems. All model performance and computing time are assessed and compared with each other. In the end, the paper offers some insights about how to choose the proper model given graph descriptive parameters such as LFR mixing parameter.

For the proposed approach in [259], eight large scale datasets are analayzed through 10 different methods for community detection. Along with the annotated communities from graph metadata, each graph dataset is associated with 11 types of community partition results. After calculating 36 types of structural properties of each filtered community from all 11 community partition results, the paper takes these structural properties as features to predict their related community detection methods via a Support Vector Machine (SVM) model. The final comparative analysis gives researchers clear insights for community detection model selection.

[260] formulates a generalized procedure of community detection. A procedure-oriented framework for benchmarking is proposed with four core modules including Setup, Detection Framework, Diagnoses, and Evaluation, which enables researchers to evaluate and compare various approaches through a universal pipeline. Particularly, in the Detection Framework, the paper shows an example by re-implementing 10 community detection models; in the Diagnoses module, key factors and key steps in each individual community detection model can be analyzed for improving model performance; in the Evaluation module, models can be assessed from multiple perspectives including efficiency, accuracy, effectiveness, and sensitivity.

[261] evaluates the top five graph matrices utilized in spectral clustering methods which can generate the best community partition. The five matrices conducted in the comparative study include adjacency matrix, standard Laplacian matrix, normalized Laplacian matrix, modularity matrix and correlation matrix. By employing k-means method on top of the decomposed eigenvectors of all matrices from a set of benchmark graphs, the paper shows that the normalized Laplacian matrix and correlation matrix achieve better performance in terms of community partition accuracy.

2.5.2 Metrics

The ill-defined community detection task leads to no universal standard to quantify the appropriateness of a community detection algorithm. [15] introduces a bunch of metrics as benchmarks from both community structure and community evaluation perspective. It also categorizes the metrics for overlapping and non-overlapping purposes. In this section, we follows the taxonomy offered by [15] to group several widely used metrics into two categories: community structure (to only describe generated communities' characteristics) and community evaluation (to compare the detected communities with ground truth communities). While I won't separate metrics based on whether it is designed for overlapping communities or not because many metrics are eligible

Metric	**Definition**
Separability	$f(c) = \frac{\lvert E_c^{in}\rvert}{\lvert E_c^{out}\rvert}$
Density	$f(c) = \frac{2\lvert E_c^{in}\rvert}{N_c(N_c-1)}$
Clustering coefficient	$f(c) = \frac{T_c^{clo}}{T_c^{clo}+T_c^{op}}$
Cut Ratio	$f(c) = \frac{\lvert E_c^{in}\rvert}{N_c(N-N_c)}$
Conductance	$f(c) = \frac{\lvert E_c^{out}\rvert}{2\lvert E_c^{in}\rvert+\lvert E_c^{out}\rvert}$
Normalized cut	$f(c) = \frac{\lvert E_c^{out}\rvert}{2\lvert E_c^{in}\rvert+\lvert E_c^{out}\rvert} + \frac{\lvert E_c^{out}\rvert}{2(M-\lvert E_c^{in}\rvert)+\lvert E_c^{out}\rvert}$
Modularity	$f(c) = \frac{\lvert E_c^{in}\rvert}{M} - \left(\frac{\lvert E_c^{in}\rvert+\lvert E_c^{out}\rvert}{2M}\right)^2$
Surprise	$f(c) = -log\frac{\binom{F_c}{\lvert E_c\rvert}\binom{F-F_c}{M-\lvert E_c\rvert}}{\binom{F}{M}}$
Significance	$f(c) = \binom{N_c}{2}D(p_c\lvert\rvert p)$
Permanence	$f(c) = \sum_{v\in c}\left[\frac{I(v)}{E_{max}(v)} \times \frac{1}{D(v)} - (1 - c_{in}(v))\right]$
Communitude	$f(c) = \frac{\frac{\lvert E_c\rvert}{M}-\left(\frac{D_c}{2M}\right)^2}{\sqrt{\left(\frac{D_c}{2M}\right)^2\left(1-\frac{D_c}{2M}\right)^2}}$

Table 2.4: Community Structure Metrics

for both types of communities and many of the rest metrics for overlapping and non-overlapping community detection are with trivial difference.

As mentioned in Table 1.1, given a graph $G(V, E)$, $C = \{c_1, c_2, ..., c_k\}$ is the set of detected communities users generated with a specific partition model. $\Omega = \{\omega_1, \omega_2, ...\omega_k\}$ is the ground truth communities given as a prior knowledge. $N = |V| = \sum_{\omega\in\Omega} |\omega| = \sum_{c\in C} |c|$ denotes the total number of the nodes in the original graph. $M = |E|$ denotes the number of total edges in the original graph. $N_c = |c|$ denotes the number of nodes in community c.

Community Structure Metrics

The structure metrics are used to measure the community intrinsic characteristics such as how well a graph community is densely connected. There are numerous metrics existing among current works and hereby I just report several most frequently used ones. In this section, given a community c, its varied structure metrics are formulated as $f(c)$. Table 2.4 summarizes all mentioned metrics which details will be further explained in the following paragraphs.

Separability measures the ratio between the numbers of edges within or on the boundary of the community c, which is calculated as:

$$f(c) = \frac{|E_c^{in}|}{|E_c^{out}|} \tag{2.64}$$

where $|E_c^{in}|$ denotes the edges within the community c and $|E_c^{out}|$ denotes the edges on the boundary of community. Higher value indicates the community is with better coherence.

Density assumes that nodes in an optimized community should be densely connected to each other. It measures a community by considering the ratio of actual number of edges and the possible edge counts within community c.

$$f(c) = \frac{2|E_c^{in}|}{N_c(N_c - 1)} \tag{2.65}$$

where N_c is the number of nodes inside community c. Higher score means the nodes in the community are better connected.

Clustering coefficient is calculated based on the number of triplet nodes within a community c. There are two types of triplets: If there are two edges among the three nodes in the triplet, it forms a open triplet T_c^{op}; If there are three edges existing in a triplet (meaning each pair of nodes are connected), it forms a closed triplet T_c^{clo}. Clustering coefficient indicates the ratio of closed triplets in the two types of triplets, which is calculated as:

$$f(c) = \frac{T_c^{clo}}{T_c^{clo} + T_c^{op}} \tag{2.66}$$

Cut ratio calculates the ratio of boundary edges out of all possible edges that leaving the community c, which is calculated as:

$$f(c) = \frac{|E_c^{out}|}{N_c(N - N_c)} \tag{2.67}$$

where N denotes the number of nodes in the whole graph. $N_c(N - N_c)$ calculates the number of possible node pairs that the community c reaching to outside. Small ratio indicates the same-community nodes are likely to be grouped together and apart from the rest nodes in the graph.

Conductance measures the ratio of edges that community c points to the inside and outside of the community. It indicates the degree of edges leaving out of current community. Higher value indicates the community c has more edges connected with external communities. The conductance is calculated as:

$$f(c) = \frac{|E_c^{out}|}{2|E_c^{in}| + |E_c^{out}|} \tag{2.68}$$

The reason that $|E_c^{in}|$ is calculated twice is because both the start and end nodes of these edges are inside the community c.

Normalized cut is an extension of conductance by adding another term related to the outside community linkages, which is calculated as:

$$f(c) = \frac{|E_c^{out}|}{2|E_c^{in}| + |E_c^{out}|} + \frac{|E_c^{out}|}{2(M - |E_c^{in}|) + |E_c^{out}|} \tag{2.69}$$

The first term in Normalized cut is the same as conductance, which represents the tendency that nodes in community c link inside the community. Similarly, the second term represents the tendency that nodes in outside communities connect to current community c. M is the total number of edges in the graph.

Modularity, as mentioned in previous chapters, is a type of scoring function to measure how well the current community surpasses random-generated communities in terms of the within-community node connection. It is calculated as:

$$f(c) = \frac{|E_c^{in}|}{M} - \left(\frac{|E_c^{in}| + |E_c^{out}|}{2M}\right)^2 \tag{2.70}$$

Derived from Modularity, **Surprise** [262, 263] also compares the distribution of the nodes and edges in a community to a random-emerged null model. By calculating the actual probability of a community given a partition result, it measures how unlikely that distribution happens. The calculation for each community Surprise score is showed as:

$$f(c) = -log\frac{\binom{F_c}{|E_c|}\binom{F-F_c}{M-|E_c|}}{\binom{F}{M}} \tag{2.71}$$

where $F = \frac{N(N-1)}{2}$ is the maximum possible number of edges in the graph G. $F_c = \frac{N_c(N_c-1)}{2}$ is the maximum possible number of edges appeared in the community c.

Significance [264] converts the original community partition task to a new thought of finding dense subgraphs with a certain number of internal edges in a random graph. The Significance score of a community with size N_c and density p_c is calculated as its log-likelihood probability of generating it from the full graph with N nodes and density p, which is rigorously proved as:

$$f(c) = \binom{N_c}{2}\mathcal{D}(p_c||p) \tag{2.72}$$

where $\mathcal{D}(\cdot||\cdot)$ denotes the Kullback-Leibler divergence.

Permanence [265, 266] is a vertex-based community quality metric to quantify how well the nodes will be persisted in the community instead of pulled out by other communities. For each node v in community c, its in-community connections $I(v)$ is divided by its maximum connections $E_{max}(v)$ to each external communities. To ensure the final score in a range of 0 to 1, it is normalized by the node degree $D(v)$. $c_{in}(v)$ denotes the internal clustering coefficient of node v in community c, which calculates the ratio of its edges within the community to all its edges. It functions as a penalty part in the permanence metric. Finally, the community score is the sum of all its nodes scores, which is calculated as:

$$f(c) = \sum_{v \in c}\left[\frac{I(v)}{E_{max}(v)} \times \frac{1}{D(v)} - (1 - c_{in}(v))\right] \tag{2.73}$$

Communitude [267] quantifies the community c in terms of the fraction of the number of

Metric	Definition
Purity	$f(C,\Omega) = \frac{1}{N}\sum_k \max_j N_{c_k,\omega_j}$
Precision (P)	$f(C,\Omega) = \frac{TP}{TP+FP}$
Recall (R)	$f(C,\Omega) = \frac{TP}{TP+FN}$
F_β	$f(C,\Omega) = \frac{(\beta^2+1)P\times R}{\beta^2(P+R)}$
Rand index	$f(C,\Omega) = \frac{TP+TN}{TP+FP+FN+TN}$
Normalized mutual information	$f(C,\Omega) = \frac{I(C,\Omega)}{[H(C)+H(\Omega)]/2}$

Table 2.5: Community Evaluation Metrics

edges within the community using the Z-score as follows:

$$f(c) = \frac{\frac{|E_c|}{M} - \left(\frac{D_c}{2M}\right)^2}{\sqrt{\left(\frac{D_c}{2M}\right)^2\left(1-\frac{D_c}{2M}\right)^2}} \tag{2.74}$$

where D_c is the sum of node degrees in community c.

Community Evaluation Metrics

Once the community partition result C is retrieved by employing a specific detection model, its performance can thereafter be evaluated by comparing with ground truth community Ω. In this dissertation, various types of measurements are introduced to help quantify model performances, which are named as community evaluation metrics and showed in Table 2.5. The score of evaluation metric given a community partition C and ground truth Ω is formulated as $f(C,\Omega)$.

To calculate **Purity**, a detected community c is assigned to a label which appears most frequently on the nodes within community c. It measures the ratio of nodes which labels are correctly assigned in each community:

$$f(C,\Omega) = \frac{1}{N}\sum_k \max_j N_{c_k,\omega_j} \tag{2.75}$$

where $N_{c_k,\omega_j} = |c_k \cap \omega_j|$ denotes the number of common nodes detected in the two communities $c_k \in C$ and $\omega_j \in \Omega$. However, purity is really sensitive to the number of communities for the original graph. If the number of communities increases, the purity score is more likely to increase as well. Therefore, it is not an ideal evaluation metric to judge whether the community partition is good or bad, especially when the number of communities is unknown.

Similarly to the evaluation in classification tasks, evaluation in community detection also builds up a confusion matrix to measure the pairwise node community relationship under four categories. A true positive (TP) node pair means two nodes in the same ground-truth community are also assigned to the same detected community. A true negative (TN) node pair means two nodes in different ground-truth communities are also assigned to different detected communities. A false positive (FP) node pair means two nodes in different ground-truth communities are mistakenly assigned to the same detected community. And a false negative (FN) node pair means two nodes in the the same ground-truth community are mistakenly assigned to different detected communities. Using the four types of node pairs, by computing all possible node pairs in the graph G, the following metrics can be retrieved including Precision, Recall, F_β and Rand index.

Precision indicates the correctness ratio of all node pairs which are predicted in the same community:

$$f(C, \Omega) = \frac{TP}{TP + FP} \tag{2.76}$$

Recall shows the ratio of same-community node pairs in the ground truth is actually retrieved in the detected communities:

$$f(C, \Omega) = \frac{TP}{TP + FN} \tag{2.77}$$

F_β is the weighted harmonic mean of Precision and Recall to balance the two metrics with a weighting factor β:

$$f(C, \Omega) = \frac{(\beta^2 + 1)P \times R}{\beta^2(P + R)} \tag{2.78}$$

Rand index considers the ratio of correctly predict node pairs in all the possible pairs:

$$f(C, \Omega) = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.79}$$

Normalized mutual information (NMI) measures the model performance through entropy perspective, which is defined as:

$$f(C,\Omega) = \frac{I(C,\Omega)}{[H(C)+H(\Omega)]/2} \tag{2.80}$$

where $I(C,\Omega)$ refers to the mutual information of the predicted community partition C with the ground truth community partition Ω. Mutual information calculates how much similar the two partitions look like, which is calculated as:

$$I(C,\Omega) = \sum_k \sum_j \frac{N_{c_k,\omega_j}}{N} log \frac{N_{c_k,\omega_j}}{N_{c_k} N_{\omega_j}} \tag{2.81}$$

And $H(\cdot)$ is the entropy function, defined as:

$$H(C) = -\sum_k \frac{N_{c_k}}{N} log \frac{N_{c_k}}{N} \tag{2.82}$$

NMI is always a number between 0 and 1. However, it can't correctly handle the case where detected community size is hugely different with ground truth community size. One of its variant, rNMI [268], solves this problem by comparing the NMI of two community partitions with a random community partition.

2.5.3 Summary

In this section, evaluation works are summarized into two tracks including comparative studies and metric introduction. The comparative studies always run empirical experiments on different graphs using various community detection models. These researches aim to find the best parameter settings for each model and explore the correlation between models and the intrinsic graph structures. As all mentioned comparative studies in this section are the best known works published in the recent decade, it gives a solid and authentic insight for the model understanding and selection.

For metric introduction, I introduce a set of structure metrics to reveal community self-characteristics. Several evaluation metrics are also introduced to assess the model performance by comparing the detected communities with the ground truth. The structure metrics such as density and modularity are classic ones and have been widely used for years in numerous research works. While some other structure metrics such as supervise and permanence are just recently designed which are still need to be testified in future works. As for those mentioned evaluation metrics, they are derived from classification evaluation metrics, which are all fundamental ones for all types of evaluation tasks.

CHAPTER 3
PERSONALIZED COMMUNITY DETECTION

3.1 Introduction

Community detection is an important topic in graph mining. By learning node community labels on the graph, we are able to detect node hidden attributes as well as explore the closeness between nodes [3]. Conventional methods are mostly user-independent to detect communities solely relying on graph topological structure [4], generate semi-supervised communities with node constraints [76], or select top-K sub graphs as user-centric communities [111]. These approaches are no longer enough to satisfy users with a pursuit of personalization, which makes involving user need into community detection to become an inevitable task.

First, user-independent approaches solely consider graph topological structure without user need. Second, semi-supervised approaches detect communities restricted by pre-selected seed nodes. As different user needs refer to different seeds, each individual user requires a separate process to run the whole model completely to get personalized communities, which is inapplicable in real cases. Third, sub-graph selection approaches only generate communities from the partial graph instead of the whole one.

To detect personalized communities on the whole graph, I propose a **g**enetic **P**ersonalized **C**ommunity **D**etection (gPCD) model with an offline and an online step. Specifically, in the offline step, I convert the user-independent graph community to a binary community tree which is encoded with binary code. Subsequently, a deep learning method is utilized to learn low-dimensional embedding representations for both user need and nodes on the binary community tree. In the online step, I propose a genetic tree-pruning approach on the tree to detect personalized communities by maximizing user need and minimizing user searching cost simultaneously. The whole genetic approach runs in an iterative manner to simulate an evolutionary process and generate a number of partition candidates which are regarded as "chromosomes" in each genetic generation. Through the selection, cross-over and mutation process, successive chromosomes are bred as better personalized community partitions to meet with user need.

3.2 genetic Personalized Community Detection

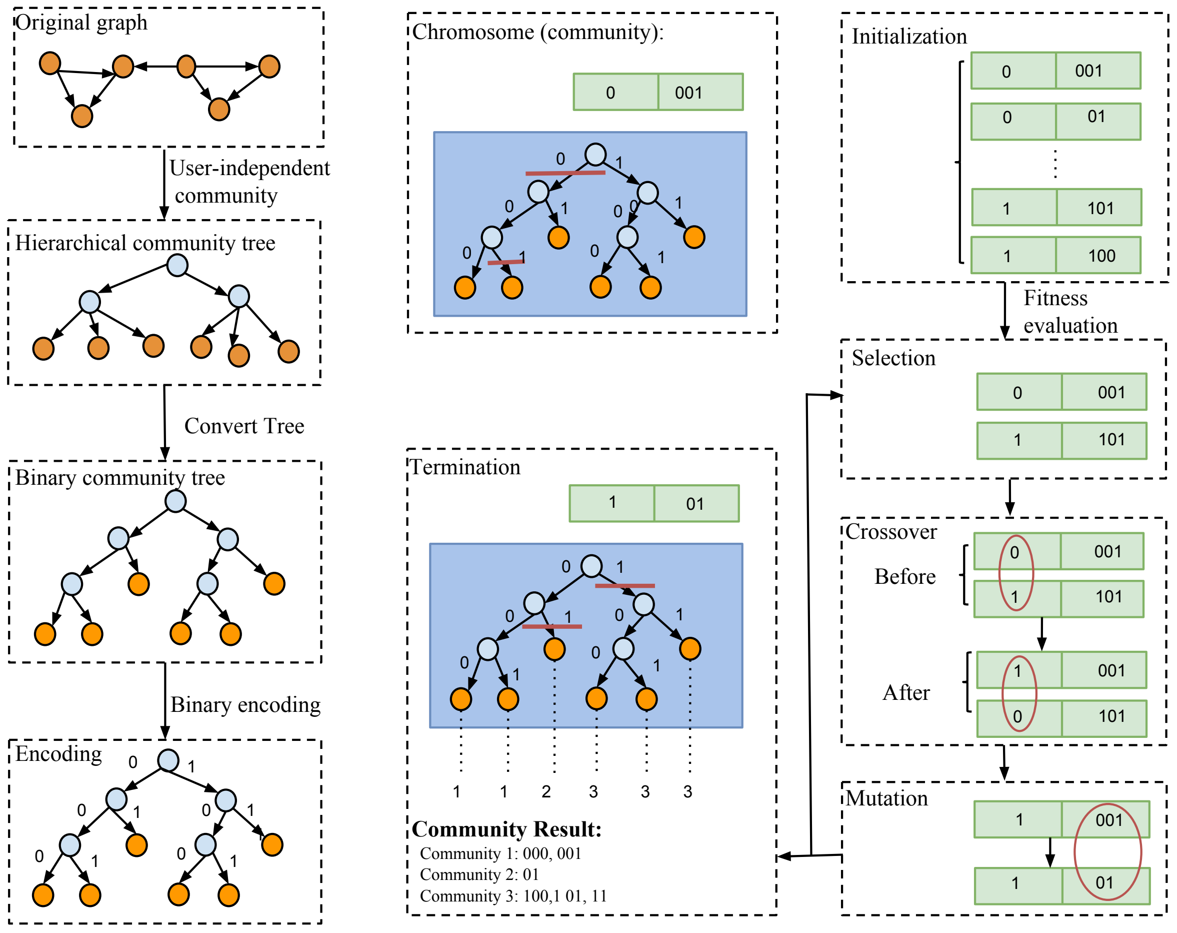
My proposed gPCD model contains an offline and an online step. Figure 3.1 shows the pipeline of the whole framework. The offline step first encodes the user-independent binary community tree (*Section 3.2.1*), and subsequently learns embedding representations for both user need and nodes on the binary community tree (*Section 3.2.2*). The online step introduces the genetic personalized community detection approach (*Section 3.2.3*). To accelerate the running speed, a distributed version of gPCD model is also deployed on HDFS (*Section 3.2.4*). To disambiguate the notations mentioned in this section to better explain the gPCD model, some commonly used notations can be found in Table 3.1.

3.2.1 Offline Community Tree Index

One challenge of solving personalized community detection problem is the computational cost due to the complexity of personalization and graph structure. In order to reduce the online workload, most of computation cost is put into the one-time offline step whose time cost can be excluded from the online personalized community detection step. Thus, I first convert the graph into a binary community tree offline to retain user-independent community information.

Infomap algorithm [210] is employed to generate the user-independent communities solely based on the graph *G(V, E)*. Infomap algorithm simulates a random walker wandering on the graph and indexes the description length of his / her random walk path via multilevel codebooks. By minimizing the description length based on the map equation below, community structures are formed for the graph.

$$L(\pi) = \sum_{i}^{m} q_{\curvearrowright}^{i} H(\mathcal{Q}) + \sum_{i=1}^{m} p_{\circlearrowright}^{i} H(\mathcal{P}^{i}) \tag{3.1}$$



(a) Offline:
binary community tree

(b) Online:
genetic pruning (Community size K = 3)

Figure 3.1: The framework of gPCD model. (a) refers to the offline construction step on the original graph and (b) refers to the online genetic pruning step to generate personalized communities.

where $L(\pi)$ is the description length for a random walker under current community partition π. $q^i_{\curvearrowright}$ and $p^i_{\circlearrowright}$ are the jumping rates between communities 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. Followed by this equation to partition communities into sub-communities , a hierarchical community tree $T_c(N^c, L^c)$ is constructed from the original graph $G(V, E)$. In $T_c(N^c, L^c)$, each parent node can have multiple child nodes which can be regarded as a community partition on the parent node. For instance, a node $N^c_k \in N^c$ from $T_c(N^c, L^c)$ represents a community of vertices. Its m child nodes $\{N^c_{k_1}, N^c_{k_2}, ..., N^c_{k_m}\}$ represent m sub-communities of vertices from $G(V, E)$ where we have $\bigcap_{i=1}^{m} N^c_{k_i} = \varnothing$ and $\bigcup_{i=1}^{m} N^c_{k_i} = N^c_k$.

In order to achieve an efficient personalized community detection in the following online step, I convert the hierarchical community tree $T_c(N^c, L^c)$ to a binary community tree $T_b(N^b, L^b)$ for index. Specifically, for m child nodes of a parent node N^c_k, a bottom-up approach is proposed to merge a selected pair of sibling nodes as a new node in an iterative manner. The approach runs until all m child nodes merged together to form the parent node N^c_k. To avoid an unbalanced tree where small communities are always left to merge with huge communities in the end, I first select the node with the smallest community size among all sibling nodes in each merging step. It is merged with its sibling node with the largest normalized linked weight (Please refer to Figure 3.1(a)). The normalized linked weight function $w(\cdot)$ between two nodes N^c_i and N^c_j is defined as:

$$w(N^c_i, N^c_j) = \frac{N^c_i \odot N^c_j - \frac{\mathcal{D}(N^c_i)\cdot\mathcal{D}(N^c_j)}{2|E|}}{|N^c_i||N^c_j|} \tag{3.2}$$

where $N^c_i \odot N^c_j$ denotes the number of edges linked between vertices in node N^c_i and N^c_j, which can be interpreted as the linkage strength between them; $|N^c_i|$ is the number of vertices inside node N^c_i; $\mathcal{D}(N^c_i)$ is the out-degree of node N^c_i (the total number of edges linked to other nodes) and $|E|$ is the total number of edges in the original graph $G(V, E)$. $\frac{\mathcal{D}(N^c_i)\cdot\mathcal{D}(N^c_j)}{2|E|}$ denotes the random linkage strength between node N^c_i and N^c_j. The $w(\cdot)$ function calculates how much that two nodes are better connected beyond random connection and is normalized by node size. Given the node N^c_i with the smallest community size and all its sibling node set S, The merging step can be formulated as:

$$\begin{aligned} N^c_j &\Leftarrow \operatorname*{argmax}_{N^c_j \in S} w(N^c_i, N^c_j) \\ N^c_* &= N^c_i \bigcup N^c_j \end{aligned} \tag{3.3}$$

The bottom-up process will stop until all child nodes are merged together to form the parent node. In the end, the hierarchical community tree $T_c(N^c, L^c)$ is fully converted to a binary community tree $T_b(N^b, L^b)$ with user-independent community information. The node size $|N^b|$ as well as the link size $|L^b|$ in $T_b(N^b, L^b)$ is at most $2|V|$ which is smaller than the size of original graph $G(V, E)$. If I consider to form the binary community tree with only k levels, the size of $T_b(N^b, L^b)$ can be even smaller.

Notations	**Descriptions**
$G(V, E)$	Original graph G with vertex set V and edge set E
$T_c(N^c, L^c)$	The hierarchical community tree generated from graph $G(V, E)$ with node set N^c and link set L^c. Each node $N^c_k \in N^c$ denotes a group of vertices belonging to V.
$T_b(N^b, L^b)$	The binary community tree reconstructed from $T_c(N^c, L^c)$. Each node $N^b_k \in N^b$ denotes a group of vertices belonging to V.
B	The binary codebook for $T_b(N^b, L^b)$. Particularly, $B_k \in B$ denotes the binary code of both $N^b_k \in N^b$ and $L^b_k \in L^b$ where L^b_k is the link points to node N^b_k.

Table 3.1: Commonly used notations in gPCD model

For running time analysis, calculating normalized linked weight takes constant time. In each merging step, node pair selection takes linear time. Therefore, in the worst case, the time complexity of binary community tree construction is $O(|V|^2)$ where the depth of the hierarchical community tree $T_c(N^c, L^c)$ is 1 and each vertex in $G(V, E)$ forms a single-vertex community.

To encode the nodes and links on $T_b(N^b, L^b)$ as binary code, the root node is encoded as 'null' first. For a parent node N^b_k with its left child node $N^b_{k_l}$ and right child node $N^b_{k_r}$, the binary code of a child node and the related link defined in the Notation Table 3.1 is calculated as:

$$B_{k_i} = \begin{cases} B_k + \text{“0”}, & i = \text{“}l\text{”} \\ B_k + \text{“1”}, & i = \text{“}r\text{”} \end{cases} \tag{3.4}$$

For instance, if the node N^b_k is with binary code “00,” its left child node's binary code is “000” while the right child node's binary code is “001.” The link L^b_k that points to N^b_k also has the binary code “00”.

3.2.2 Community and User Need Representation

Node2vec [16] helps to learn fixed-length embeddings for both user need and communities. It simulates random walks on the graph $G(V, E)$ and learns the vertex embedding by optimizing the sequential relationships from random walk paths. In the end, each vertex V_k in graph $G(V, E)$ has a vector representation as $\vec{V_k}$. Each node N_k^b on the binary community tree $T_b(N^b, L^b)$ refers to a vertex community C_k in the graph $G(V, E)$. Its representation $\vec{C_k}$ is calculated as the averaged embedding of all vertices inside the community. In the end, the binary community tree $T_b(N^b, L^b)$ represents the hierarchical community partition of Graph $G(V, E)$. Each node N_k^b on the tree is indexed with three attributes: a group of vertices from graph $G(V, E)$, a binary code B_k, and an embedding representation $\vec{C}_k$.

On the other hand, User need (query) I can also be represented by a combination of t different vertices $\{V_1, V_2 ... V_t\}$ in the graph $G(V, E)$. In this study, two different scenarios for user need representation are offered:

Vertex-based Query. User need can be directly represented by the vertices based on the generation probability $P(V_k|I)$ between them. Hence the user need representation $\vec{I}$ is calculated as:

$$\vec{I} = \sum_{k=1}^{t} P(V_k|I) \cdot \vec{V_k} \tag{3.5}$$

For instance, in a music sharing network, each vertex V_k denotes a music and a user listing history can be used to reflect the user need I. $P(V_k|I)$ therefore can be regarded as the probability that a music being listened by the user.

Text-based Query. Under this scenario, user need I is represented as a text query, and each vertex V_k in the graph $G(V, E)$ also contains textual content. From language model viewpoint, each vertex importance weight is the query likelihood $P(I|V_k)$, and the user need can is the weighted average of vertex embedding:

$$\vec{I} = \frac{\sum_{k=1}^{t} P(I|V_k) \cdot \vec{V_k}}{\sum_{k=1}^{t} P(I|V_k)} \tag{3.6}$$

In either case, user need is conceptualized as an embedding with the same dimension as the node embeddings on the binary community tree. It enables very efficient online personalized community detection in later steps. And running Node2vec takes most of the time in this step.

3.2.3 Online Genetic Pruning

The whole process, as the Figure 3.1 shows, is to generate communities by pruning the constructed binary community tree. After each cut on a link, the original tree will be separated into two subtrees. After a specific number of cuts to the links on the tree, a fixed number of communities with different resolutions are detected. By applying genetic selection, crossover, and mutation steps, the model converges to the optimized solution efficiently with a clear-defined fitness function. The details are shown in the following paragraphs.

Genetic Representation

A chromosome is formed by a set of genes $\{g_1, g_2, ..., g_{K-1}\}$, and each gene g_i holds a cut link L_i^b in the binary community tree $T_b(N^b, L^b)$. Since communities can be created by cutting links on the offline tree, a chromosome can be represented as a generated community partition of the original graph $G(V, E)$ in this way. To constrain a chromosome so that it can be decoded to a fixed number of communities, four **Cutting Rules** are necessarily to be applied:

- **Rule 1**: If a link L_i^b is picked to cut on the binary community tree $T_b(N^b, L^b)$, its pointing node N_i^b will be retrieved and all the vertices within it form a community.
- **Rule 2**: If a link L_i^b and its ancestor link L_j^b are stored in the same chromosome, all vertices in L_i^b's related node N_i^b are a subset of vertices in L_j^b's related node N_j^b. In this case, the two cut links generate two communities where community C_i is all vertices in N_i^b and community

C_j is the remaining vertices in N_j^b but not in N_i^b. It can be formulated as $C_i = \bigcup_k \{V_k | (V_k \in N_i^b)\}$ and community $C_j = \bigcup_k \{V_k | (V_k \in N_j^b) \cap (V_k \notin N_i^b)\}$.

- **Rule 3**: Sibling links can't be stored in the same chromosome, and it is not allowed to store duplicated links in a chromosome.
- **Rule 4**: The depth's upper bound is set to be d, which means all eligible cut links should be located in the first d depth on the binary community tree. It avoids to generate super tiny communities and hugely reduces the genetic searching scope on cut links.

By applying the cutting rules to the online pruning process, I ensure a K community partition can be retrieved from a chromosome with $K - 1$ cut links.

Initialization

Initially, the model generates a given number P chromosomes as the seed "chromosome population". And each iteration in the genetic approach breeds a new "generation" of chromosome population. In order to ensure a chromosome is an encoder of a K community partition, $K - 1$ links will be randomly picked (on the binary community tree) following the cutting rules and stored in the related genes of a chromosome.

Fitness Function

As each chromosome can be decoded as a community partition, it is important to measure the quality of each generated chromosome (how well the generated communities can satisfy user need). The measurement is hosted in a fitness function.

In the proposed model, the fitness function simulates the user searching behavior on the graph given the community partition. For instance, a user can be more likely to pick the most relevant communities while avoiding the redundant information already selected. With the help of the offline step, the relevance score of node N_i^b (community C_i) towards user need I can be calculated with the cosine similarity $cos(\vec{I}, \vec{C_i})$, and the information redundancy can be $\sum_{C_j \in S_c} cos(\vec{C_j}, \vec{C_i})$

where S_c is the set of communities that the user have already picked from the communities decoded from the target chromosome. Following this, I use a greedy selection approach to iteratively rank and pick communities given a chromosome (community partition) until all communities are picked:

$$\underset{C_i}{\operatorname{argmax}}\, \lambda \cdot cos(\vec{I}, \vec{C_i}) - (1-\lambda) \cdot \frac{\sum_{C_j \in S_c} cos(\vec{C_j}, \vec{C_i})}{|S_c|} \tag{3.7}$$

where C_i is the candidate community to be picked and $|S_c|$ is the number of communities already been picked. λ is a parameter controls whether user prefers to obtain new useful information or to avoid redundant information.

For chromosome quality evaluation, a query-generated vertex ranking list l_q is first created by retrieving top n vertices relevant to the query (user need) with the largest cosine similarities on embeddings of graph $G(V, E)$. I store the top n vertex ranking label $R(l_q) = \{1, 2, ..., n\}$ as the pseudo ground truth. On the other hand, given the k_{th} chromosome ch_k in the current chromosome generation, I can also retrieve the community-generated ranking of each vertex $V_k \in l_q$ from the sequentially selected communities decoded by the chromosome. I assign the ranking label on each vertex V_k based on the following formula:

$$\sum_{V_j \in l_q} \Phi(\delta(V_j) < \delta(V_k)) + 1 \tag{3.8}$$

V_j refers to all vertices in l_q. $\delta(V_j)$ shows the ranking (selection sequence) of the community which V_j belongs to. Φ is a binary operator to determine whether V_j satisfy the condition $\delta(V_j) < \delta(V_q)$. This formula helps to construct the community-generated ranking label $R(l_c)$. For instance, when $n = 3$, I have a query-generated ranking list $l_q = \{V_1, V_2, V_3\}$ and its related ranking label $R(l_q) = \{1, 2, 3\}$. Given a chromosome where the community of V_1 and V_2 is the same and selected before V_3, I can generate the related community ranking label $R(l_c) = \{1, 1, 3\}$ with the same vertex sequence of l_q.

Then, I define the fitness function $f(\cdot)$ to evaluate the chromosome ch_k. As I have the query-generated ranking label $R(l_q)$ (ground truth) and community-generated ranking label $R(l_c)$ from ch_k, I calculate their Kendall's τ correlation coefficient as the fitness score $f(ch_k)$ of chromosome ch_k where higher score means the chromosome ch_k can generate better personalized communities to meet with user need.

$$f(ch_k) = 1 - \frac{\sum_{i=1}^{n} R(l_{ci}) \cdot R(l_{qi})}{\sum_{i=1}^{n} R(l_{ci})^2 \cdot \sum_{i=1}^{n} R(l_{qi})^2} \tag{3.9}$$

where $R(l_{ci})$ is the i_{th} vertex ranking in community-generated ranking label $R(l_c)$ and $R(l_{qi})$ is the i_{th} vertex ranking in query-generated ranking label $R(l_q)$. Kendall's τ is a widely used metric to evaluate the correlation between two lists where higher score means stronger correlation. Thus, higher fitness score reflects that the generated community ranking $R(l_c)$ can better meet with user need $R(l_q)$.

Moreover, it is clear that the fitness function aims to separate all top n vertices in different communities to get the optimal case. It matches the research goal to generate high resolution communities on vertices which are more relevant to user need. As the number of community is a given number K, it also leads to a coarser manner partition on the remaining less relevant vertices. On the other hand, the binary community tree $T_b(N^b, L^b)$ and the Cutting Rule 4 naturally preserve the community structure and unite the most relevant vertices in the same community. Hence the whole genetic approach is a gambling process. The final chromosome result is the equilibrium case to detect communities both contain graph topological structure and meet with user need.

Selection

I select the superior chromosomes from current chromosome population based on their fitness scores. The probability that the k_{th} chromosome ch_k is picked can be calculated via the Softmax normalization function $p(ch_i) = \frac{exp(f(ch_k))}{\sum_{i=1}^{P} exp(f(ch_i))}$. Then, the Fitness Proportionate Selection method is applied to randomly select P chromosomes into chromosome pairs based on probability distribution. In order to enhance optimization efficiency, I also use elitism selection to ensure the best chromosome in the current generation will always be selected to the next generation.

Crossover

To reach global optimum community partition efficiently, given a pair of chromosomes, the crossover operation can randomly exchange part of the genes in both chromosomes to produce a new pair of chromosomes with a certain crossover rate.

In order to make sure that the newly generated chromosomes meet the cutting rules, an **Exchange Rule** is defined to restrict gene exchange: If gene g contains link L_g^b, g can't do crossover process with genes that contain either link L_g^b or its sibling link $L_g^{b'}$. This rule can help avoid having duplicated links or sibling links stored together in the newly generated chromosome (To satisfy Cutting rule 3).

After m random numbers are selected from $\{1, 2, ..., K-1\}$ as exchanged gene position indexes, genes located in the chosen positions of two chromosomes will exchange the stored link restricted by the Exchange Rule.

Mutation

Mutation operation is applied to avoid local optimization. If a chromosome is chosen to mutate, a gene within the chromosome will be randomly picked, and its stored link will be changed to another link restricted by the Exchange Rules. An example is illustrated in Figure 3.1(b) where the link stored in the second gene is changed from “001” to “01”.

Termination

After T iterations, the whole process stops and the current best chromosome is retrieved as the final result. Choosing the number of T is dependent on the task. In order to decode the final chromosome to the related community partition, all genes in the chromosome are sorted in an ascending order based on the binary code of their stored cut link. Vertices whose binary codes start with the same cut link's binary code will be assigned to the same community label. And its later assigned community label can overwrite the previous assigned community label. For instance, if there are a vertex with binary code “0011” and two cut links with binary code “00” and “001”, the vertex will be assigned to a community label “00” first, and its community label is overwritten by “001” afterwards. The Termination step in Figure 3.1(b) also illustrates a vivid example. In this way, the binary code of the binary community tree can help to decode the final chromosome into communities in an efficient way.

3.2.4 Distributed gPCD

To enhance the online step efficiency, a MapReduce framework is utilized to enable the distributed genetic evolution. Figure 3.2 depicts the personalized community detection under a MapReduce framework. The chromosome collection is either originally initialized from binary community tree or obtained from the last generation. It contains the whole chromosome population in the central depository. In its first “Splitter” process, all chromosomes are split into M groups based on their hash values and sent out to related M Mappers to calculate the “Fitness” scores. In the same Mapper, after all chromosomes are assigned fitness scores, based on their scores, a Combiner groups all chromosomes together and random select equal number of chromosomes with duplicated as the “Selection” step. All the selected chromosomes are sent to R reducers (I set $R = M$ arbitrarily in order to better represent time complexity) to form pairs for the “Crossover” and “Mutation” step, calculate new chromosome offsprings for the next generation and store them back to the central repository.

The complexity of the proposed algorithm is $O(2^d KP)$ without parallel computing and $O(\frac{2^d KP}{M})$ with parallel computing, where d denotes the upper bound where the cut links are restricted in the top d depth of the binary community tree $T_b(N^b, L^b)$; K denotes the community number; P denotes the initialized population size of the genetic algorithm and M denotes the number of Mappers/Reducers in parallel environment. As all the parameters are considerably small (compared with the node/edge size in the original graph), the whole process runs very fast to retrieve the final community partition.

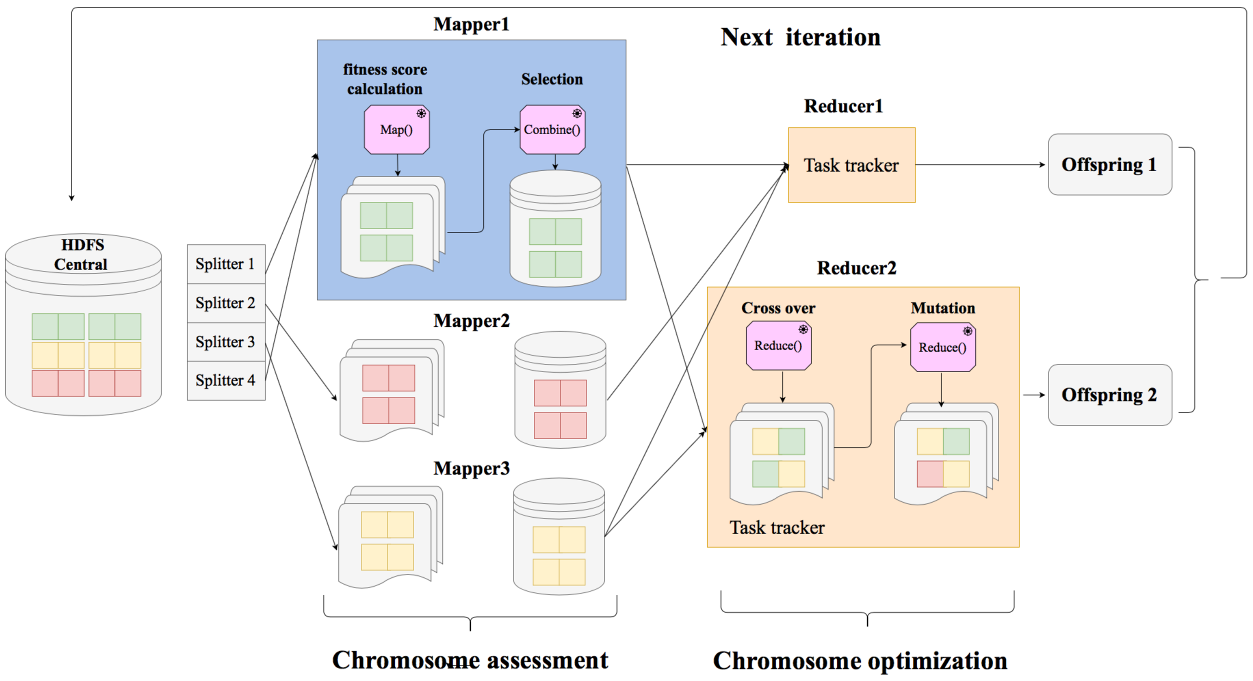


Figure 3.2: Online parallel computing process on Hadoop Distributed File System (HDFS)

3.3 Experiments

3.3.1 Datasets

Datasets Description

Dataset	Node description		Edge description	
	Type	Size	Type	Size
Scholarly	paper	166,170	citation	750,181
Music	song	145,203	co-listening	1,172,525

Table 3.2: Dataset Description

Table 3.2 shows the statistics of the two datasets. The scholarly graph is unweighted and directed, while the music graph is a weighted and undirected.

Scholarly Graph: It contains academic publications with metadata extracted from ACM Digital Library. From the dataset, I build the experimental graph via paper citation relationship. Each vertex in the graph represents a paper, and if a paper cites another paper, there will be an edge linking the two. My model aims to detect personalized communities on the scholarly graph for authors.

For each author in the dataset, I represent his/her need in two ways: their previous publications (text-based query) and cited paper history (vertex-based query).

Music Graph: It contains user listening histories and user-generated playlists from an online music streaming service, Xiami. I create a music graph with songs as the vertices and co-listening relationship as the edges. If two songs appear in the same user's listening history, there will be an edge linking them two. For users in the music dataset, I represent their music tastes (user need) from the songs in their listening history.

Ground Truth Construction

The ground truth for the two datasets are generated based on each user's publishing/ citing/ listening history:

For the scholarly dataset, the references of 112 random sampled papers are manually annotated from their literature reviews where authors summarize previous works. Different paragraphs (or sub-sections) in the literature review typically focus on separate but coherent topics while the same paragraph talks about the same topic. Based on this assumption, the papers cited in the same paragraph/sub-section naturally form a community with high resolution. To ensure each paper's cited papers form enough communities and each community contains enough papers, only papers with no fewer than three topics and all of whose communities have at least five papers are kept. After applying all these filters, 101 papers are left for evaluation.

For the music dataset, each user has several self-generated playlists. The songs in each playlist should contain a coherent theme. To avoid the playlists sharing mutually exclusive themes with other playlists, a Jaccard similarity check is applied on any pair of playlists created by the same user. If a user has at least two highly correlated playlists (Jaccard coefficient between them is above 0.5), I remove one of the playlists. Each playlist forms a separate community. Furthermore, to ensure the number of playlist and playlist size are both large enough, only users with at least three playlists and each playlist contains at least five songs are kept. In the end, there are 117 users who meet the above criteria.

In this paper, as all communities constructed in the ground truth are relevant communities with high resolution for users, my task is generating personalized communities to reconstruct the ground truth on two different datasets with vertex- and text-based user need.

3.3.2 Settings

Metrics & Parameter Settings

F1-score (F1), Rand index (Rand), Jaccard index (Jaccard) and running time are reported as the evaluation metrics in this paper. Based on empirical studies, population size P is 100. Crossover rate is 95%. Mutation rate is 1%. The maximum depth of binary community tree d is 10. The number of iteration T is 30. User searching preference λ is 0.6. Community size K is 50. The number of Mappers/ Reducers for parallelization M is 50. The number of top vertices to construct pseudo ground truth n is 10. Parameters in Infomap and Node2vec are both the default settings in their original papers.

Baselines

Considering both efficacy and efficiency, I select eight widely used user-independent community detection models. Ideally, to achieve personalized community detection, user-independent models should run on each user separately by assigning higher weights on user related edges. Thus, their time complexity should be only compared with my online step time complexity as my offline step is independent with user numbers. In this paper, to run baselines within acceptable time, I report their user-independent community results as the average performance.

- **Spinglass** [128]: Spinglass constructs communities by minimizing the Hamiltonian score on signed graphs.[1]
- **Fast Greedy** (FG) [269]: Fast Greedy is a greedy search method to get the maximized modularity for community detection.[2]

[1]https://github.com/antoine-lizee/SG
[2]https://github.com/kjahan/community

- **Louvain** [133]: Louvain is an agglomerative method to construct communities in a bottom-up manner guided by modularity.[1]

- **Walktrap** [270]: Walktrap detects communities based on the fact that a random walker tends to be trapped in dense part of a network.[2]

- **Infomap** [210]: Infomap generates communities by simulating a random walker wandering on the graph and indexing the description length of his random walk path via multilevel codebooks.[3]

- **Bigclam** [80]: Bigclam generates overlapping communities via a non-negative matrix factorization approach.[4]

- **DeepWalk** [271]: DeepWalk generates node embeddings via random walks and utilizes K-means on node embeddings to detect communitis.[5]

- **Node2vec** [16]: Node2vec is an extended version of DeepWalk with a refined random walk strategy.[6]

3.4 Discussion

3.4.1 Evaluation Results

There are two scenarios to construct user need. For the scholarly graph, including a citation (vertex-based query) model and a keyword (text-based query) model. In the citation model, for each user, we first extract all the papers he/she cited before, then use their centroid embedding as user need vector $\vec{I}$. In the keyword model, for each user (author), I first extract all keywords he/she

[1]https://github.com/taynaud/python-louvain
[2]https://www-complexnetworks.lip6.fr/ latapy/PP/walktrap.html
[3]http://www.mapequation.org/code.html
[4]https://github.com/snap-stanford/snap/tree/master/examples/bigclam
[5]https://github.com/phanein/deepwalk/tree/master/deepwalk
[6]https://github.com/aditya-grover/node2vec

used in all previous papers to form a text query. Then I retrieve the top 100 relevant papers given the query based on probability language model with Dirichlet smoothing. Finally, I average those retrieved papers' vectors as the user need vector $\vec{I}$.

Model	**Scholarly Graph**			**Music Graph**		
	F1	Rand	Jaccard	F1	Rand	Jaccard
Spinglass	0.4294	0.4149	0.3593	0.4282	0.5317	0.2823
FG	0.4290	0.3852	0.3735	0.4645	0.4100	0.3070
Louvain	0.4417	0.4546	0.3627	0.1832	0.4201	0.1174
Walktrap	0.4304	0.3777	0.3777	0.3999	0.3507	0.3490
Infomap	0.4436	0.4165	0.3606	0.2147	**0.6074**	0.1344
Bigclam	0.2314	0.2572	0.1348	0.1499	0.2078	0.1227
DeepWalk	0.3904	0.3237	0.3234	0.3535	0.3253	0.3001
Node2vec	0.4001	0.3472	0.3433	0.4122	0.4101	0.3087
gPCD-Citation	**0.5351***	**0.4551***	**0.4086***	-	-	-
gPCD-Keyword	0.5069	0.4114	0.3708	-	-	-
gPCD-Listening	-	-	-	**0.5188***	0.5865	**0.3550***

Note: "*" means the p-value through a pairwise t-test is smaller than 0.001.

Table 3.3: Personalized Community Evaluation on gPCD and Baselines.

For the music task, text information is not available. The centroid embedding of the songs listened to by a target user are taken as user need.

Community Accuracy: I compare the average performance of my model on all testing users with baselines. Table 3.3 shows the detailed metrics. When running on the Scholarly graph, both Citation model and Keyword model can achieve around 10% increase on F1-score compared with all baselines. Citation model also performs the best in Rand Index and Jaccard Index. For Music graph, my Listening model also has a significant improvement on F1-score and Jaccard Index. Although it has similar performance on Rand Index compared with Infomap, I believe my model in fact works much better due to the Infomap's poor performance on the rest two metrics. Moreover, I apply pairwise t-tests for all metrics on all testing users. All metrics' p-values in Citation model and the p-values of F1-score and Jaccard Index in Listening model are all smaller than 0.001, which means the improvements of my model performance are significant compared with baselines.

Running Time Comparison: Table 3.4 shows both the theoretical time complexity and real running time. To represent baseline algorithms' time complexity, "V" refers to the vertex set and "E" refers to the edge set in the graph $G(V, E)$. For some models (FG, Walktrap, and Infomap.), their specific time complexities are officially mentioned in the original papers. The time complexity of Louvain and Bigclam are roughly estimated in the original papers as well but those papers don't mention specific numbers. For Spinglass, DeepWalk and Node2vec, I can't find the exact time complexity in existing studies. Hence in this paper, I arbitrarily assign labels based on the their real running speed. Considering the running time, all the baseline algorithms run relatively fast except for the Spinglass algorithm. However, compared with all other models, the distributed gPCD always performs the fastest. Its real running time of is less than one-tenth of the fastest baseline's running time.

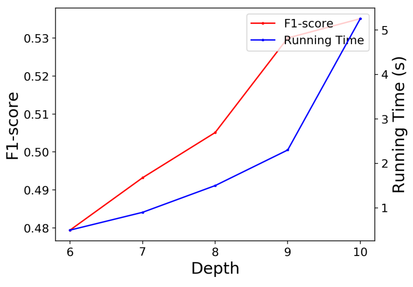
Model	**Time Complexity**	**Scholarly Graph** (s)	**Music Graph** (s)
Spinglass	very slow	12548.68	10372.17
FG	$O(\|V\|log^2\|V\|)$	280.60	272.34
Louvain	linear	80.01	63.02
Walktrap	$O(\|V\|^2 log\|V\|)$	638.44	503.24
Infomap	$O(\|V\|(\|V\| + \|E\|))$	501.79	425.63
Bigclam	linear	57.01	112.43
DeepWalk	fast	720.56	688.32
Node2vec	slow	3508.44	3100.12
gPCD	$O(\frac{2^d KP}{M})$	**5.25**	**6.50**

Table 3.4: Running time analysis on gPCD and all baselines in seconds (s).

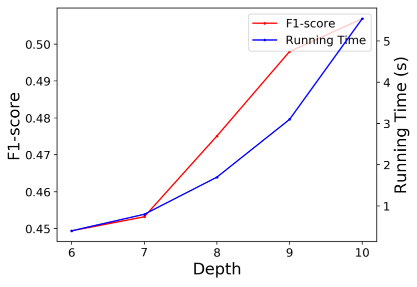
3.4.2 Parameter Analysis

I show how three parameters can affect my gPCD model performance in this section. They are the depth of the binary community tree d, genetic iteration number T and user searching preference λ. Figure 3.3 show the overall impacts of all tuned parameters.

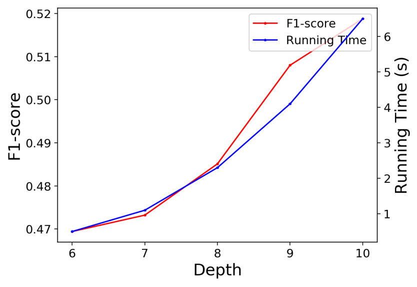
Depth on the Tree: Figure 3.3(a) to Figure 3.3(c) show how the depth of the binary community tree affects the model performance in accuracy and efficiency. From the figures, larger depth leads to a better personalized community detection result, while causes an exponential running time increase at the same time. Based on empirical studies, the upper bound of the depth is set to be 10



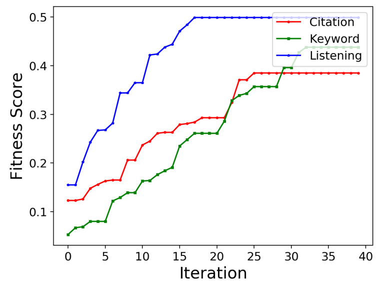
(a) Depth d in Citation model



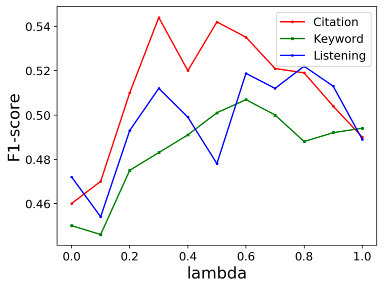
(b) Depth d in Keyword model



(c) Depth d in Listening model



(d) Iteration T in all models



(e) User searching preference λ in all models

Figure 3.3: Parameter effects on model performance

in this paper. While the depth selection may varies based on different graph sizes.

Convergence Analysis: I observe the best chromosome updates in 40 iterations. From Figure 3.3(d), I can see the fitness score start to be stable after the 30 iterations, which means the best chromosome is no longer changed after around 30 iterations. Thus, I set $T = 30$ as the default iteration number in my approach.

Searching Preference: In Figure 3.3(e), λ reflects the user searching preference whether he/she wants to explore new information or avoid redundant information. By selecting λ from 0 to 1, I find the F1-score are not very stable or have a clear correlation with λ. Based on the empirical experiments, I achieve the best performance on three models when $\lambda = 0.6$.

CHAPTER 4
CROSS-GRAPH COMMUNITY DETECTION

4.1 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 [131], spectral approach [12], dynamic approach [47] and deep learning approach [191], are able to estimate the internal/external connectivity and generate high-quality communities directly on nodes [4].

For sparse graphs, as mentioned in the Chapter 1, to deal with the lack-of-connection problem, I propose a novel research task – 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. Note that, in spit of the small sparse graphs can engage with a specific field, the main graph is quite comprehensive and noisy. As Figure 1.3 shows, not all the connections in Amazon (shopping graph) can be equally important for the two candidate app graphs. In the example, 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 study, I 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 cope with the main graph heterogeneity issue and reduce noisy information by taking care of graph local structure. Theoretically, I can offer at most $\mathcal{O}(N^3)$ user triplets to learn graph community structure while conventional community detection methods by default can only be applied on $\mathcal{O}(N)$ users (N is the number of users in the sparse graph).

Therefore, I 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 $\langle u_i, u_j, u_k \rangle$, I 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.

4.2 Method

4.2.1 Task Overview

As aforementioned, conventional methods suffer from graph sparseness problem. In this study, I 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

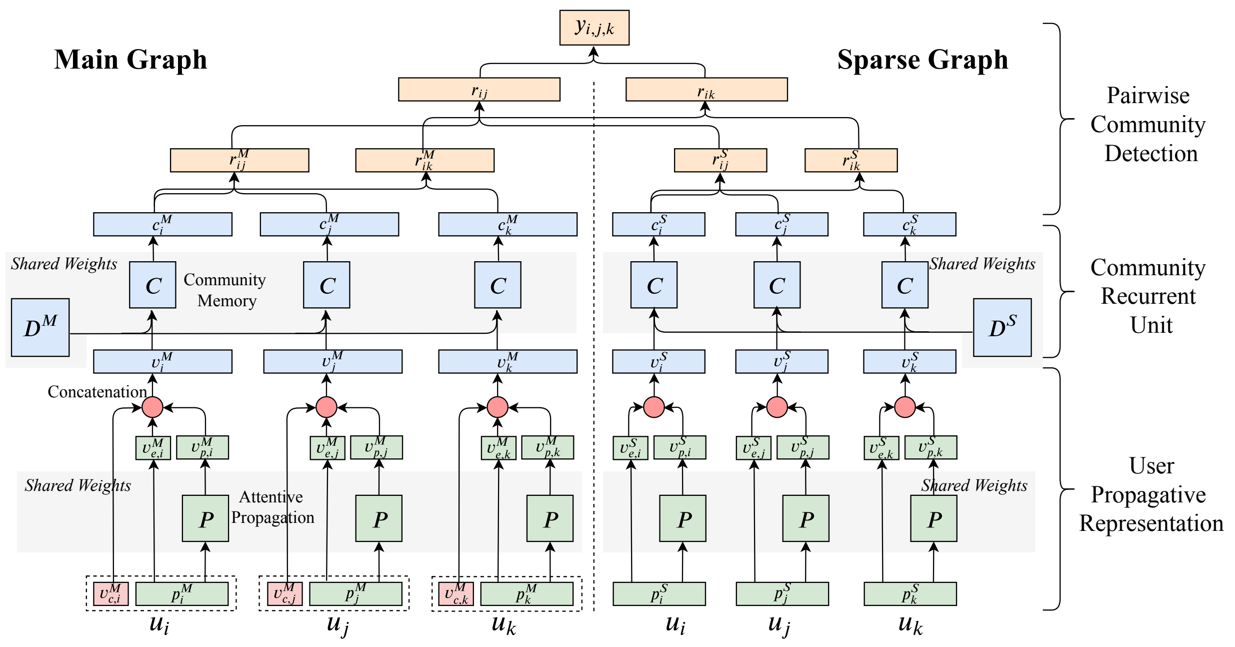


Figure 4.1: The overall architecture of my proposed PCCD model. It contains three major modules mentioned in the right part of the figure. Each training instance is a mutual user triplet $\langle u_i, u_j, u_k \rangle$. Compared with the sparse graph, the main graph involves raw user community as part of model input.

to the sparse graph to support its user community detection.

Specifically, my proposed model (showed in Figure 4.1) 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**), I 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, my proposed model detects communities firstly on separate graphs to obtain all user raw community representations (Section 4.2.2). Their propagative representations are learned through a community-level and a node-level filter (Section 4.2.3). Both representations are jointly utilized for predicting user community affiliation in each graph via a Community Recurrent Unit (Section 4.2.4). In the end, I integrate the community affiliation distributions to identify the pairwise community relationship of the triplet (Section 4.2.5). Particular training strategies are introduced in the end (Section 4.2.6).

Note that even though my 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 4.4.3 will verify its effectiveness on different types of users.

4.2.2 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 my model performance. Therefore, I 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 5.3.2, Infomap 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:

$$L(\pi) = \sum_{i}^{m} q^i_{\curvearrowright} H(\mathcal{Q}) + \sum_{i=1}^{m} p^i_{\circlearrowright} H(\mathcal{P}^i) \tag{4.1}$$

where $L(\pi)$ is the description length for a random walker under current community partition π. $q^i_{\curvearrowright}$ and $p^i_{\circlearrowright}$ are the jumping rates between communities 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.

In the end, given a user u_i, I obtain its one-hot community representation $v^M_{c,i}$ in the main graph

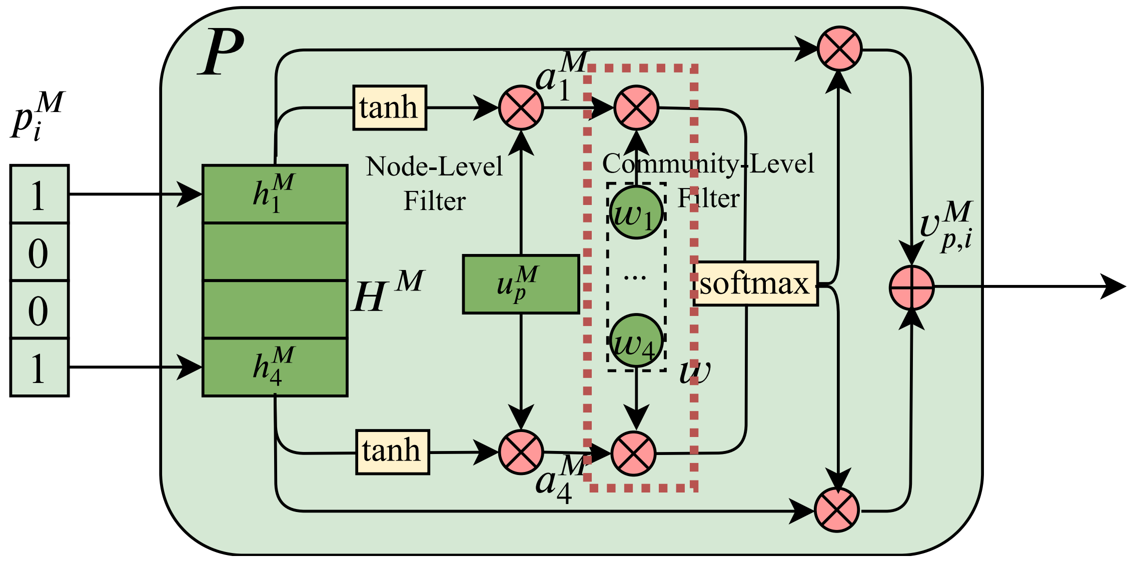


Figure 4.2: The two-level filter to support user propagative representation estimation in the main graph M. Sparse graph S does not contain the community-level filter (wrapped in the red rectangle).

M, which is regarded as part of user representations for the model input.

4.2.3 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 4.2) 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:

$$v_{e,i}^G = \tanh(W_e^G p_i^G + b_e^G) \tag{4.2}$$

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 and heterogeneous, 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 4.2.

Community-Level Filter: As the raw detected communities in the sparse graph is not reliable because of graph sparseness, I only apply the community-level filter in the main graph M. According to the raw community partition π calculated in Section 4.2.2, 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, I 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), I 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:

$$a_j^G = (u_p^G)^\intercal \tanh(W_p^G h_j^G + b_p^G) \tag{4.3}$$

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.

I 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 $\mathrm{softmax}(\cdot)$ function, the related representation of user u_i in both graphs are calculated as follows:

$$
\begin{aligned}
v_{p,i}^{M} &= \sum_{n_j \in \mathcal{N}^M(u_i)} \mathrm{softmax}(a_j^M w_{\pi(n_j)}) h_j^M \\
v_{p,i}^{S} &= \sum_{n_j \in \mathcal{N}^S(u_i)} \mathrm{softmax}(a_j^S) h_j^S
\end{aligned} \tag{4.4}
$$

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 $\mathrm{batch\_norm}(\cdot)$ function is applied on top of the concatenated representations in both graph:

$$
\begin{aligned}
v_i^M &= \mathrm{batch\_norm}([v_{c,i}^M, v_{e,i}^M, v_{p,i}^M]) \\
v_i^S &= \mathrm{batch\_norm}([v_{e,i}^S, v_{p,i}^S])
\end{aligned} \tag{4.5}
$$

4.2.4 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 4.3. 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

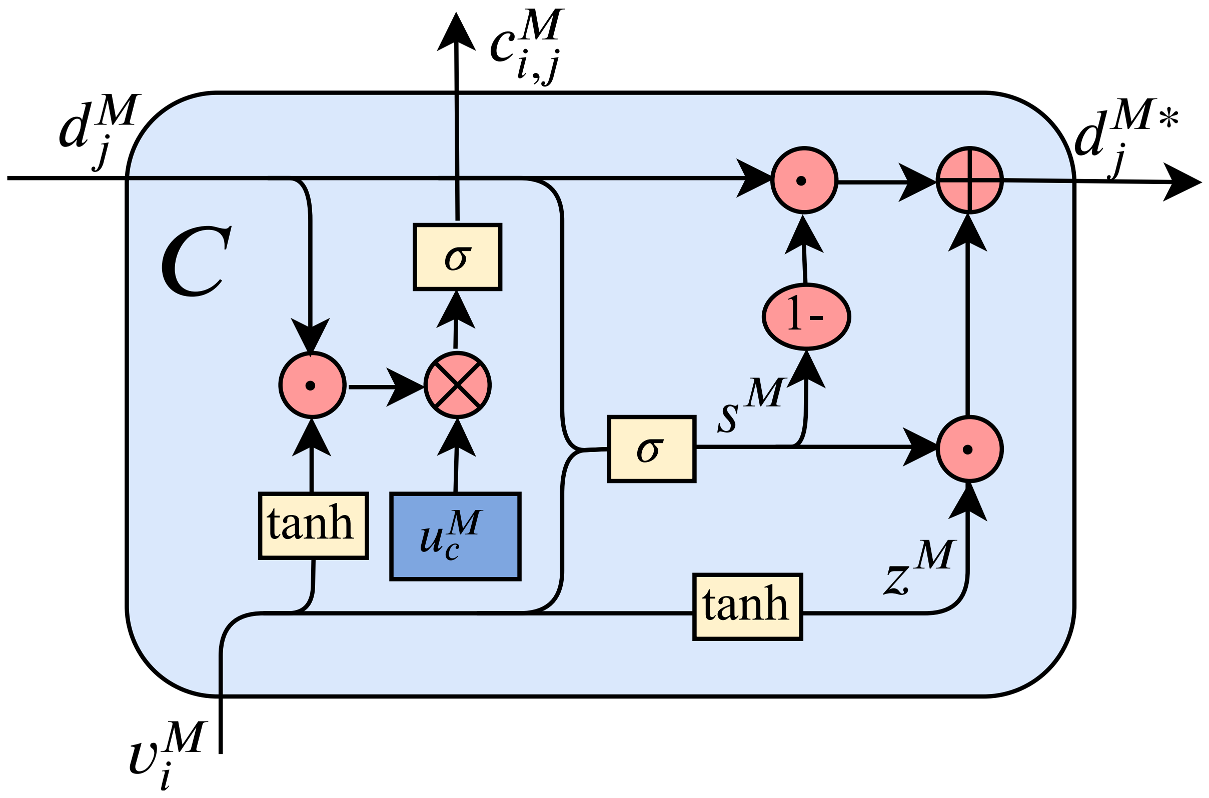


Figure 4.3: The flow of Community Recurrent Unit in the main graph M. The left part refers to the community affiliation gate and the right part is the community update gate.

score $c_{i,j}^G$ towards the j_{th} community in the graph G is calculated as follows:

$$c_{i,j}^G = \sigma((u_c^G)^\intercal(\tanh(v_i^G) * d_j^G)) \tag{4.6}$$

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, I use a delicate RNN variant, where the update process is calculated as follows:

$$
\begin{aligned}
s^G &= \sigma(W_s^G[v_i^G, d_j^G] + b_s^G) \\
z^G &= \tanh(W_z^G v_i^G + b_z^G) \\
d_j^{G*} &= (1 - s^G) * d_j^G + s^G * z^G
\end{aligned}
\tag{4.7}
$$

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 my 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:

$$
\mathcal{L}_c^G = \frac{1}{2K^2} \sum_{i,j} \cos(d_i^G, d_j^G) \tag{4.8}
$$

where K is the number of communities in each graph.

4.2.5 Pairwise Community Detection

Similar to RankNet [272], 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:

$$
y_{i,j,k} = \frac{1}{2}(1 + S_{jk}) \tag{4.9}
$$

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, I convert the pairwise ranking task to a regression task.

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

$$r_{ij}^G = W_r^G(c_i^G - c_j^G) + b_r^G \tag{4.10}$$

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

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

$$r_{ij} = \tanh([r_{ij}^S, r_{ij}^M]) \tag{4.11}$$

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

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

$$\hat{y}_{i,j,k} = \sigma(W_o(r_{ij} - r_{ik}) + b_o) \tag{4.12}$$

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:

$$\mathcal{L}_{total} = \underbrace{-y_{i,j,k} log\hat{y}_{i,j,k} - (1 - y_{i,j,k}) log(1 - \hat{y}_{i,j,k})}_{optimization} + \underbrace{\alpha(\mathcal{L}_c^S + \mathcal{L}_c^M)}_{constraint} \tag{4.13}$$

4.2.6 Training Strategy

Although my 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 my model to improve model robustness, including shared weights and masked training.

First, as my model should be insensitive to the sequence of input user triplet, the weight matrices, biases and project vectors in both Section 4.2.3 and Section 4.2.4 should be shared among the three users. Moreover, in Section 4.2.4, 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. 4.7) from input triplets.

Second, to alleviate the negative effects of excessive connections in main graph M, I 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.

4.3 Experiments

4.3.1 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, I 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 my model robustness with respect to the graph size, two cross-graph datasets are constructed in different scales, which are showed in Table 4.1. 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 4.1: 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, I 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 my ground truth data, I run all baseline models (will be introduced in Section 5.3.2) on both seven-day sparse graphs which are assumed to contain sufficient information for detecting user communities. Only mutual user communities agreed by all seven baselines are kept, from which mutual user triplets are subsequently selected. In this paper, I define three types to label pairwise community closeness for a mutual user triplet, including "*closer*", "*similar*", and "*farther*" (detailed definition is in Section 4.2.1). 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, my 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.

4.3.2 Baselines and Settings

As there is neither reliable nor open-sourced studies on pairwise cross-graph community detection, I have no direct comparative models. In this paper, I select one random model and six compromised but state-of-art baselines which are originally designed for whole graph community detection. In the first seven baselines, after user communities are detected, I calculate user triplet labels via Eq. 4.9. 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 my model. Here is the details of how these baselines calculate user communities:

- **random**: users are randomly assigned to a given number of communities.
- **Infomap** [210]: A random walk based method to detect user communities by minimizing description length of walk paths.[1]
- **DNR** [140]: A novel nonlinear reconstruction method to detect communities by adopting deep neural networks on pairwise constraints among graph nodes.[2]
- **LSH** [273]: A random walk based method for community detection in large-scale graphs.[3]
- **Node2vec** [16]: 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** [80]: A non-negative matrix factorization method to detect overlapping communities in large scale graphs.[4]
- **WMW** [274]: 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 my model with learning rate as 0.01. Community number on both graphs is $K = 50$ (Eq. 4.8). Community constraint weight is $\alpha = 0.1$ (Eq. 4.13).

[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

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.

4.4 Discussion

4.4.1 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 my model can only be tested by utilizing both graphs. Table 4.4 shows the overall model performance under three graphs. Particularly, I run my model ten times and report the average performance result. To verify my model's superiority, I calculate the performance differences between my 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 4.2.5). 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.

Table 4.4 shows that 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. 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 my 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.

4.4.2 Ablation Study

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

In this section, I aim to explore whether all components of my 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). I 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 4.2.

Among the six components, if I 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 my 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 I 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 4.2.3.

4.4.3 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 4.3: Three types of user performance in two datasets.

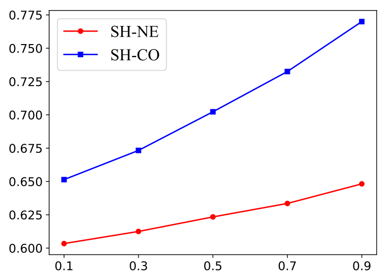
As aforementioned in Section 4.3.1, there are three types of users in my cross-graph datasets including *MU* (mutual users appeared in both graphs), *MO* (users only appeared in the main graph), and *SO* (users only appeared in the sparse graph). Unlike my training data which is constructed only with mutual users, my testing data can be with all types of users. To examine my model performance on each type of users, user triplets with only single type of users are screened, i.e., an eligible user triplet only contains three *MO* type users. In fact, the performance on *MO* user triplets reflects the capability of my model to solve cold-start problem as these users have no behaviors in the sparse graphs. The result showed in Table 4.3 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 my model on *SO* user triplets is still better than most of baselines running on the combined graph.

4.4.4 Graph Sparsity Influence

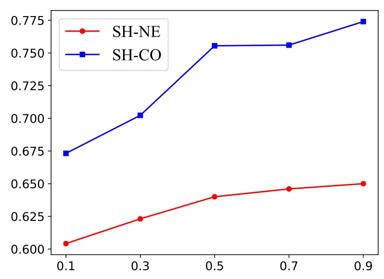
To explore my model robustness on graph sparsity, I randomly keep δ ratio of the total links in the sparse graph, where $\delta = \{0.1, 0.3, 0.5, 0.7, 0.9\}$. My model is trained on the whole main graph and each truncated sparse graph. The reported evaluation metrics under all sparse graphs with varied scales are visualized in Figure 4.4. 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 4.4. 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.

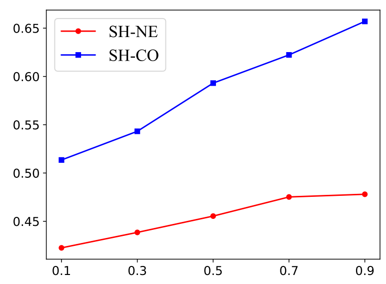
Although classification related metrics significantly increase with more links preserved (i.e., F1 score in SH-CO dataset increases 10% when involving sparse graph), the retrieval related metrics (MRR, NDCG and MAP) do not change much in the mean time. One possible reason is that the



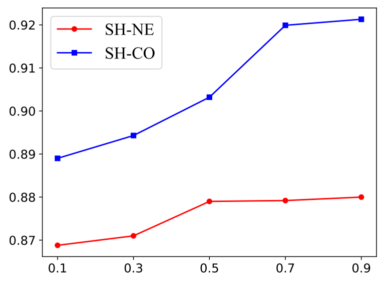
(a) ACC



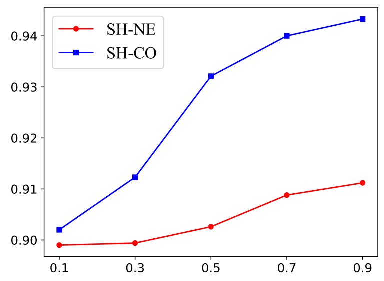
(b) F1



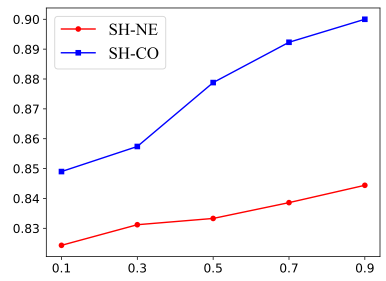
(c) MCC



(d) MRR



(e) NDCG



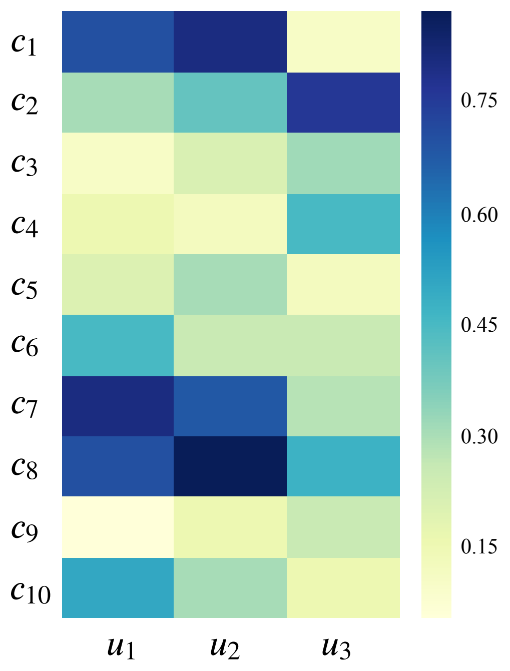
(f) MAP

Figure 4.4: 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.

information from the main shopping graph already contains enough information for the pairwise ranking in user triplets. For example, without considering sparse graph information, my 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.

4.4.5 Case Study

In order to testify whether my proposed Community Recurrent Unit is able to accurately calculate user affiliation scores in each community, I 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 4.5. 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, I 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 4.5.



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 4.5: 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.

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 4.5, 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 my calculated community affiliation distribution is consistent with actual user behaviors, which indicates my Community Recurrent Unit (CRU) is functional well to reflect user community information.

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

CHAPTER 5
COMMUNITY-AWARE DYNAMIC PRODUCT SUMMARIZATION

5.1 Introduction

Prior investigations on review summarization, mainly follow Natural Language Generation (NLG) approaches, such as [275] uses new Recurrent Neural Network (RNN) variant that uses gated connections to construct a character-level text generation mode and [276] designs a multi-task model to predict rating and generate review summarization simultaneously using pairwise user-product relationship. [277] uses a memory network for review summarization generation.

However, all these models are originally designed for static review summarization and take product reviews as model input. They are vulnerable to depict product characteristic changes because of (real-time review) data sparsity. For instance, after investigating 2.16 billion products sold by *Taobao*, a world-leading online shopping website owned by Alibaba, only 0.05% of products are able to gather more than 100 reviews within a three-day window. Thus, review-based approaches are not feasible for dynamic summarizations in large scope because of the lack of instant reviews.

On the other hand, user behavior offers an alternative to address characteristic dynamics. Based on the statistics of the *Taobao* collection, more than 2.53% of products can receive more than 100 multi-type user behaviors (e.g., *'Click'* or *'Purchase'*) within a three-day window where the coverage is 50 times greater than the review scope. Rational Choice Theory [278], on the theory side, proves that user shopping behavior rationality has a coherent relationship with the product peculiarity.

Motivated by all aforementioned scenarios, I propose a **B**ehavior based **D**ynamic **S**ummarization (BDS) model to accommodate user behavior for dynamic product summarization with the help of community detection techniques. The user shopping preference is stable in a relatively long-term period [279], which offers us the theoretical feasibility to learn product behavior representation

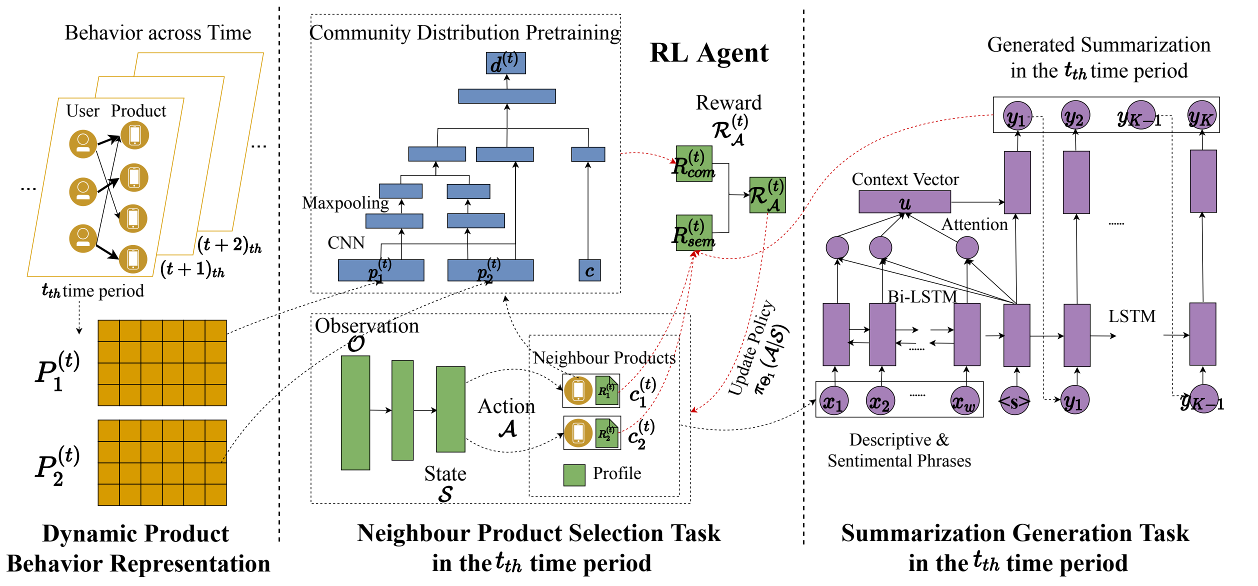

Behavior across Time
User
Product
$(t+2)_{th}$
$(t+1)_{th}$
t_{th} time period
$P_1^{(t)}$
$P_2^{(t)}$
Dynamic Product Behavior Representation
Community Distribution Pretraining
$d^{(t)}$
Maxpooling
CNN
$p_1^{(t)}$
$p_2^{(t)}$
c
Observation $\mathcal{O}$
Action $\mathcal{A}$
State $\mathcal{S}$
Neighbour Products
$c_1^{(t)}$
$c_2^{(t)}$
Profile
Neighbour Product Selection Task in the t_{th} time period
RL Agent
Reward $\mathcal{R}_{\mathcal{A}}^{(t)}$
$R_{com}^{(t)}$
$R_{sem}^{(t)}$
$\mathcal{R}_{\mathcal{A}}^{(t)}$
Update Policy $\pi_{\Theta_1}(\mathcal{A}|\mathcal{S})$
Generated Summarization in the t_{th} time period
y_1 y_2 y_{K-1} y_K
Context Vector
u
Attention
Bi-LSTM
LSTM
x_1 x_2 x_w <s> y_1 y_{K-1}
Descriptive & Sentimental Phrases
Summarization Generation Task in the t_{th} time period

Figure 5.1: The overall architecture of BDS model in the t_{th} time period. Each color refers to an individual model component. Dynamic Product Behavior Representation (for all products) and Community Distribution Pretraining (for seed products only) are optimized before training the main multi-task model (Neighbor Product Selection & Summarization Generation). Red dashed lines show the workflows to calculate RL reward and in return to optimize RL policy. Data Prerequisite steps are omitted here for simplicity but are illustrated in detail in the main paper.

from user dynamic behavior and consistent shopping preference. The learned representation supports neighbor product selection from a group of seed products with abundant instant reviews (**Task 1**) and meanwhile implicitly helps to generate summarization from product own descriptive phrases and neighbor products' filtered sentimental phrases (**Task 2**). As both user behavior and seed products' instant reviews are changed across time, the generated summarizations and neighbor products from detected communities are associated with changes as well.

5.2 BDS Model

The overall model architecture is sketched in Figure 5.1. My final goal is to generate product dynamic summarizations using behavior data instead of product reviews. To optimize BDS model, the involved data are categorized into two groups: training & validation data and seed product data. The training & validation data contains products with both sufficient user behavior and temporal reviews in an individual time period. While seed product data are products required to have sufficient behaviors and reviews across all time periods. The training & validation data helps to calculate dynamic product behavior presentations (*Section 5.2.2*) and seed product data pretrains community distribution model (*Section 5.2.3*). Subsequently, a multi-task model dynamically selects neighbor products (*Section 5.2.3*) from seed products whose sentimental phrases filtered from reviews contribute to generate product summarization (*Section 5.2.4*).

5.2.1 Data Prerequisite

Before optimizing BDS model, four following types of information need to be clarified and extracted as data prerequisite:

Seed Products: The top products having the most sufficient behaviors and reviews across all t time periods are regarded as seed products. Their community distributions are prior-known in each time period, which are later used to guide neighbor product selection (Section 5.2.3).

Product Descriptive Phrase: AliNLP[1], a paid NLP service by Alibaba, can extract name entities and sentimental phrases from E-commerce review contexts. Each product is associated with a description profile. After applying AliNLP Named Entity Recognizer (NER) on their profiles, I only keep the noun phrases to characterize related products.

Review Sentimental Phrase: The AliNLP sentiment analyzer can also extract sentimental phrases related to specific aspects from seed product reviews, i.e., ‘poor quality’ is a sentimental phrases of ‘*Quality*’ aspect. Later on, these filtered sentimental phrases will be concatenated with product descriptive phrases as the model input for product summarization (Section 5.2.4).

Product Community Distribution: In this paper, seed product’ community distribution is prior knowledge which is learned from their reviews via AliNLP. As user behaviors has strong correlation to product characteristics, their behavior based communities should also be strongly correlated with communities calculated from product reviews which depicts product characteristics. It is a normalized vector in which each dimension represents the product affiliation towards each

[1]English Tutorial: https://www.alibabacloud.com/help/product/57736.html
Chinese Tutorial: https://data.aliyun.com/product/nlp

community, It is the ground truth to guide Community Distribution Pretraining (Section 5.2.3).

5.2.2 Dynamic Product Behavior Representation

Matrix factorization is utilized on dynamic user behaviors to learn product behavior representation across all time periods. In the initial time period, I learn both user and product m types of behavior representations via related user behavior $\{B_1^{(0)}, ..., B_m^{(0)}\}$, where a behavior type can refer to '*Click*' or '*Purchase*', etc. The i_{th} type of user behavior $B_i^{(0)}$ is in a format of sparse matrix where each row denotes a user and each column denotes a product. The data points in $B_i^{(0)}$ denote users' i_{th} type of behaviors on products. Considering all potential methods listed in *Section 5.3.2*, I empirically select Probabilistic Matrix Factorization (PMF) to decompose $B_i^{(0)}$ to user representation matrix $U_i^{(0)}$ and product representation matrix $P_i^{(0)}$ by satisfying the following equation:

$$B_i^{(0)} = U_i^{(0)}(P_i^{(0)})^\intercal, i = 1, ..., m \tag{5.1}$$

As user shopping preference is consistently stable, once user representation $U_i^{(0)}$ is calculated from the initial period, it remains unchanged and bridges product dynamic behavior representation in subsequent time periods.

In the t_{th} time period, the matrix form of Least Squares Approximation [280] helps to calculate the i_{th} type of product behavior representation $P_i^{(t)}$ given related user behavior $B_i^{(t)}$ and user consistent representation $U_i^{(0)}$:

$$\begin{aligned}
&B_i^{(t)} = U_i^{(0)}(P_i^{(t)})^\intercal, i = 1, ..., m \\
&\Rightarrow (U_i^{(0)})^\intercal(B_i^{(t)} - U_i^{(0)}(P_i^{(t)})^\intercal) = 0 \\
&\Rightarrow (U_i^{(0)})^\intercal U_i^{(0)}(P_i^{(t)})^\intercal = (U_i^{(0)})^\intercal B_i^{(t)} \\
&\Rightarrow (P_i^{(t)})^\intercal = ((U_i^{(0)})^\intercal U_i^{(0)})^{-1}(U_i^{(0)})^\intercal B_i^{(t)} \\
&\Rightarrow P_i^{(t)} = (B_i^{(t)})^\intercal U_i^{(0)}((U_i^{(0)})^\intercal U_i^{(0)})^{-1}
\end{aligned} \tag{5.2}$$

5.2.3 Neighbor Product Selection Task

Community Distribution Pretraining

This pretraining process is only leveraged on seed products. In the t_{th} time period, I combine product behavior representation and category to estimate product community distribution via a hybrid CNN-MLP (Convolutional Neural Network and Multilayer Perceptron) approach. Let c be the category of a target product, stored originally as one-hot embedding. To better represent its enriched information, I first apply an one-layer MLP with $\mathrm{tahn}(\cdot)$ activation function to convert it as a dense vector v^c.

$$v^c = \mathrm{tahn}(W^c c + b^c) \tag{5.3}$$

where W^c and b^c denote the weight matrix and bias respectively.

For the same product, I can also obtain its multi-type behavior representation $\{p_1^{(t)}, ..., p_m^{(t)}\}$, where $p_m^{(t)} \in P_m^{(t)}$ denotes its m_{th} type behavior representation. As CNN kernel can help to filter out the most important dimensions (local features) from a vector, a CNN layer $\mathrm{cnn}(\cdot)$ with Max pooling mechanism $\mathrm{max\_pooling}(\cdot)$ is applied to capture product i_{th} type local feature representation $l_i^{(t)}$.

$$l_i^{(t)} = \mathrm{max\_pooling}(\mathrm{tahn}(\mathrm{cnn}(p_i^{(t)}))), i = 1, ..., m \tag{5.4}$$

Subsequently, I average all m types of local feature representation to a single vector $l^{(t)}$. Similarly, I calculate the average product behavior vector $p^{(t)}$ as global feature representation of m types of behaviors:

$$\begin{aligned} l^{(t)} &= \frac{1}{m}\sum_{i=1}^{m} l_i^{(t)} \\ p^{(t)} &= \frac{1}{m}\sum_{i=1}^{m} p_i^{(t)} \end{aligned} \tag{5.5}$$

In the end, a product has three types of information representation, including global feature representation $p^{(t)}$, local feature representation $l^{(t)}$, and category representation v^c. I concatenate all these information to calculate the estimated product community distribution $d^{(t)}$ over all pre-selected aspects via one layer MLP normalized by $\mathrm{softmax}(\cdot)$ function:

$$d^{(t)} = \mathrm{softmax}(W^d[p^{(t)}, l^{(t)}, v^c] + b^d) \tag{5.6}$$

where W^d and b^d are related weight matrix and bias. $[\,,]$ denotes the concatenation operation.

This pretraining process is optimized by minimizing the cross entropy between the estimated product community distribution $d^{(t)}$ and the actual product community distribution calculated from product reviews (Section 5.2.1). This optimization step has to be done ahead of the main multi-task model introduced in the following sections. In the later training process, for all products in the training & validation dataset, their estimated community distribution can be calculated from the optimized pretraining model.

Reinforcement Neighbor Selection

In the t_{th} time period, among h seed products with sufficient reviews, a policy gradient approach earns an action to select s neighbor products $\mathcal{A} = \{\alpha_1^{(t)}, \alpha_2^{(t)}, ..., \alpha_s^{(t)}\}$ out of h candidates where $\alpha_s^{(t)}$ denotes the s_{th} selected neighbor product. As I do not consider the sampling sequence on neighbor products, the reinforcement approach is a one-step Markov Decision Process (MDP) with single state $\mathcal{S}$ and single action $\mathcal{A}$.

Assume there are n products in total across all time periods, given a target product, the initial observation $\mathcal{O}$ is the product itself, represented as a one-hot embedding $\in \mathbb{R}^n$. The state $\mathcal{S}$ is learned via a two-layer Multilayer Perception (MLP) on the initial observation $\mathcal{O}$:

$$\mathcal{S} = \mathrm{softmax}(W_2 \mathrm{tahn}(W_1 \mathcal{O} + b_1)) \tag{5.7}$$

where W_1,W_2 and b_1 denote related weight matrices and bias, respectively.

The learned state $\mathcal{S} \in \mathbb{R}^h$ is the selection probability distribution over h seed products. Assuming an action is taken to sample s neighbor products with related probability weights

$\{\omega_1^{(t)}, \omega_2^{(t)}, ..., \omega_s^{(t)}\} \in \mathcal{S}$, the sampling policy $\pi_{\Theta_1}(\mathcal{A}|\mathcal{S})$ can be therefore calculated as:

$$\pi_{\Theta_1}(\mathcal{A}|\mathcal{S}) = s! \prod_{i=1}^{s} \omega_i^{(t)} \tag{5.8}$$

Θ_1 denote the parameters to be learned. The factorial of s ($s!$) denotes the number of permutations for the selected neighbor products as the neighbor products are sequence insensitive. $\prod_{i=1}^{s} \omega_i^{(t)}$ is the generative probability of each permutation.

To assess the fitness of the s selected neighbor products for the target product, I design two dynamic rewards: a community reward to measure the community similarity, and a semantic reward to calculate the content similarity between s neighbor products and the target product.

Community Reward: In the t_{th} time period, I estimate the target product's community distribution as $d_a^{(t)}$ by the Community Distribution Pretraining (Section 5.2.3). The community distributions of all its selected neighbor products $\{d_1^{(t)}, ..., d_s^{(t)}\}$ are known as prior knowledge (Section 5.2.1). To evaluate pairwise distribution similarity, Pearson correlation calculates the community reward $\mathcal{R}_{com,i}^{(t)}$ of the i_{th} selected neighbor product $\alpha_i^{(t)}$ as follows:

$$\mathcal{R}_{com,i}^{(t)} = \frac{\mathbb{E}[(d_a^{(t)} - \mu(d_a^{(t)}))(d_i^{(t)} - \mu(d_i^{(t)}))]}{\sigma(d_a^{(t)})\sigma(d_i^{(t)})}, i = 1, ..., s \tag{5.9}$$

where $\mathbb{E}(\cdot)$ denotes the expectation, $\mu(\cdot)$ denotes the mean and $\sigma(\cdot)$ denotes the standard deviation. Larger similarity score offers a higher reward to the related neighbor product.

Semantic Reward: In the t_{th} time period, the semantic reward of neighbor products is measured by the accuracy of generated product summarization. In this paper, we use the word level Jaccard Similarity as the indicator to calculate the semantic reward $\mathcal{R}_{sem}^{(t)}$ for all selected neighbor products, which contains two parts: the averaged Jaccard Similarity between all neighbor product original reviews and real product summarization, and the Jaccard Similarity between generated product summarization and actual product summarization:

$$\mathcal{R}_{sem}^{(t)} = \frac{1}{s} \sum_{i=1}^{s} \frac{|R_i^{(t)} \cap Y^{(t)}|}{|R_i^{(t)} \cup Y^{(t)}|} + \frac{|\hat{Y}^{(t)} \cap Y^{(t)}|}{|\hat{Y}^{(t)} \cup Y^{(t)}|} \tag{5.10}$$

where $R_i^{(t)}$ denotes all original reviews of the i_{th} neighbor product $\alpha_i^{(t)}$, $\hat{Y}^{(t)}$ denotes the generated summarization of target product, and $Y^{(t)}$ denotes its actual summarization. The total reward $\mathcal{R}_{\mathcal{A}}^{(t)}$ of neighbor products is the weighted sum between community reward and semantic reward controlled by a weighting factor γ:

$$\mathcal{R}_{\mathcal{A}}^{(t)} = \sum_{i=1}^{s} \omega_i^{(t)} \mathcal{R}_{com,i}^{(t)} + \gamma \mathcal{R}_{sem}^{(t)} \tag{5.11}$$

Task Optimization: I use policy gradient method to optimize the sampling policy, aiming to maximize the expected total reward for neighbor products. The expected reward $\mathcal{J}_{sel}(\Theta_1)$ in the t_{th} time period is:

$$\mathcal{J}_{sel}(\Theta_1) = \mathbb{E}_{\mathcal{A} \sim \pi_{\Theta_1}(\mathcal{A}|\mathcal{S})}[\mathcal{R}_{\mathcal{A}}^{(t)}] \tag{5.12}$$

Then, the gradient is estimated using the likelihood ratio trick:

$$\begin{aligned} \nabla_{\Theta_1} \mathcal{J}_{sel}(\Theta_1) &= \nabla_{\Theta_1} \sum_{\mathcal{A}} \pi_{\Theta_1}(\mathcal{A}|\mathcal{S}) \mathcal{R}_{\mathcal{A}}^{(t)} \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \nabla_{\Theta_1} log \pi_{\Theta_1}(\mathcal{A}_i|\mathcal{S}) \mathcal{R}_{\mathcal{A}_i}^{(t)} \end{aligned} \tag{5.13}$$

where $\mathcal{A}_i$ denotes the i_{th} of N randomly sampled actions (selecting s neighbor products from h seed products).

5.2.4 Summarization Generation Task

In the t_{th} time period, the filtered neighbor product sentimental phrases together with product own descriptive phrases are concatenated into a sequence $X^{(t)} = \{x_1, ..., x_w\}$. It is used as the input of Neural Machine Translation (NMT) model to generate product summarization sequence $Y^{(t)} = \{y_1, ..., y_k\}$. w and k denote the input and output sequence length, respectively. Filtering out most of emotional and other irrelevant words in advance can better map the input to aspect oriented summarizations instead of subjective review summaries.

The input sequence $X^{(t)} = \{x_1, ..., x_w\}$ is fed one-by-one into the encoder (a single-layer bidirectional LSTM), producing a sequence of encoder hidden states $\{e_1, ..., e_w\}$. In decoding step i, the decoder (a single-layer unidirectional LSTM) has a decoder hidden state h_i. Its context vector u_i is generated via an Attention mechanism on all encoder hidden states and current decoder hidden state:

$$\begin{aligned} a_{ij} &= \text{attention}(h_i, e_j), j = 1, ..., w \\ a^*_{ij} &= \frac{\exp(a_{ij})}{\sum_{k=1}^{w} \exp(a_{ik})} \\ u_i &= \sum_{j=1}^{w} a^*_{ij} e_j \end{aligned} \tag{5.14}$$

where a_{ij} denotes the attention weight of encoder hidden state e_j. a^*_{ij} is the normalized weight by Softmax function. The weighted sum of all encoder hidden states, u_i, is the context vector for current step i, reflecting the auxiliary information from input sequences.

A one-layer MLP is subsequently utilized on the combination of context vector u_i and decoder hidden state h_i to generate the vocabulary probability distribution P_{vocab}:

$$P_{vocab} = \text{softmax}(W^o[u_i, h_i] + b^o) \tag{5.15}$$

where W^o and b^o are related weight and bias.

In decoding step i, the generation loss for target word y_i is its negative log likelihood, $-logP_{vocab}(y_i)$. The overall generation loss $\mathcal{J}_{gen}(\Theta_2)$ is the average of all k step generation losses and Θ_2 are all related parameters to be optimized.

$$\mathcal{J}_{gen}(\Theta_2) = \frac{1}{k} \sum_{i=1}^{k} -logP_{vocab}(y_i) \tag{5.16}$$

In the multi-task model, $\mathcal{J}_{sel}(\Theta_1)$ and $\mathcal{J}_{gen}(\Theta_2)$ both need to be minimized during the model training process. Each task is learned separately and alternately after taking a certain number of training data batches in their optimization processes.

5.3 Experiments

5.3.1 Dataset

From *Taobao*, a world-leading E-commerce website owned by Alibaba, we collect user behavior and product reviews during the period from Apr/12/2018 to Jul/10/2018.the raw dataset is split into two parts: the first two-week data, as the initial time period t_0, helps to generate user consistent shopping preference (*Section 5.2.2*). For the rest data, we empirically set fifteen days as the time window to split it evenly into five consecutive time periods ($t_1 \sim t_5$). In each time period, four types of information need to be prepared and calculated in advance to support model training, including multi-type user behavior, product profile, product community distribution, and product ground truth summarization:

First, multi-type user behavior is used for generating dynamic product behavior representation (*Section 5.2.2*). Three types of user behavior are considered in this paper including '*Click*', '*Add Cart*' and '*Purchase*'.

Second, from product profile, product category is encoded as one-hot embedding for sentiment distribution prediction (*Section 5.2.3*). Descriptive phrases are extracted from product description profiles to support aspect summarization generation (*Section 5.2.4*).

Third, product community distribution is calculated from reviews by considering textual content in four common aspects including '*Quality*', '*Cost-performance Ratio*', '*Fitness*' and '*Material*'.

Fourth, in *Taobao*, a user can rate reviews with thumbs-up or thumbs-down signal. For each product in the training data, we firstly select its top ten reviews relevant to the four picked aspects and with the largest number of thumbs-ups. After that, using AliNLP sentiment analyzer, we can locate and filter out the aspect-related sentences from the reviews as the ground truth for summarization generation (*Section 5.2.4*).

In total, the whole dataset contains 125,598 products, 51,366 users and 108,749,788 multi-type behaviors. We use 80% of the data for training, 10% for validation and 10% for testing.

5.3.2 Baselines and Settings

As the main contribution of this paper lies in the reinforcement neighbor product selection task, five baselines are chosen to compare from neighbor product selection viewpoint. **1) Title Similarity** (TS): Neighbor products are selected with the shortest Levenshtein Distance on titles. **2) Random**: Neighbor products are randomly selected. **3) PMF**: PMF [281] firstly learns product embeddings via matrix factorization. Neighbor products are selected with the highest cosine similarity score on product embeddings. **4) GBPR**: GBPR [282] uses a Bayesian based collaborative filtering method to assign user preferences on products. Neighbor products are selected with the most similar user preferences. **5) EALS**: EALS [283] is a fast matrix factorization approach which learns product embeddings. Neighbor products are selected with the highest cosine similarity score on product embeddings. After the neighbor products are selected, their reviews' sentimental phrases together with product own descriptive phrases are utilized to generate product summarization via the same NMT model struture as our BDS model. We intentionally apply the same generative model so as to compare the effectiveness of neighbor products selection in different models.

To assess the usefulness of neighbor product information, our model is compared with another two generative models only leveraging product own metadata. **6) Raw Title** (RT) uses the product titles and **7) Title-Review** (TR) uses the concatenation of product title and sparse reviews as the input of the same NMT model to generate aspect summarization.

Mini-batch (*size = 20*) Adam SGD optimizer is used to train our model for 100 epochs. Learning rate is 0.01. Dimension of product & user representation is 300 (*Section 5.2.2*). $s = 5$ neighbor products are selected from $h = 100$ seed products (*Section 5.2.3*). Vocabulary size in summarization generation (*Section 5.2.4*) is *15K*. Other parameter details will be offered once the paper gets published.

5.3.3 Evaluation Metrics

In this paper, we report the model performance via the following metrics from both automatic and human perspectives. Automatic metrics evaluate the accuracy of the generated summarizations by objectively calculating the correctness of predicted words. While human metrics evaluate the semantic quality of generated summarizations by subjectively considering their sentence readability.

- **ROUGE** (RG)[1] : evaluates text summarization quality by comparing the overlap between generated sequences and the ground truth. We report RG-1, RG-2 and RG-L in this paper.
- **METEOR**[2] : is the harmonic mean of generated summarizations' unigram precision and recall. It offers stemming and synonymy matching along with standard exact word matching.
- **Human Evaluation**: We generate summaries of 200 random products by each of the seven baselines and our model. To compare BDS model with each baseline, three human votes are collected from a crowd-sourcing platform for each pair of product summaries ($200 * 7 * 3 = 4,200$ human judgements). People are required to vote the one with more comprehensive information and better sentence structure. We define *winning time rate* (WTR) and *winning count rate* (WCR) as two human evaluation metrics. Given a pair of model A and model B, the WTR of A is the ratio of winning products for A and the WCR of A is the ratio of winning votes for A. For instance, given two products α and β, if method A gets two votes for α and one vote for β, its WTR is $(1+0)/(1+1) = 0.5$ and WCR is $(2+1)/(3+3) = 0.5$.

5.4 Discussion

As neighbor product selection (Task 1) is an unsupervised approach whose ultimate goal is to support product summarization (Task 2), I only report the experimental results on Task 2 to demonstrate my model's superiority over the baselines across different timestamps. The efficacy of each input component is also presented here.

5.4.1 Automatic Evaluation

I run my model ten times and report the average evaluation results in the left part of Table 5.1. To verify my model's superiority, I calculate the performance differences between my model and each

[1]https://pypi.org/project/pyrouge/

[2]http://www.cs.cmu.edu/ alavie/METEOR/

Model	**Automatic**				**Human**	
	RG-1	RG-2	RG-L	METEOR	WTR	WCR
RT	36.19	10.01	27.95	16.05	0.00	0.01
TR	43.05	19.08	33.10	19.24	0.07	0.13
TS	42.97	18.85	32.85	19.34	0.13	0.24
Random	44.21	19.58	33.98	19.86	0.03	0.13
PMF	45.72	20.25	35.21	20.80	0.14	0.28
GBPR	45.09	19.67	34.55	20.36	0.32	0.39
EALS	44.66	19.50	34.28	20.32	0.08	0.17
BDS	**51.11***	**23.55***	**39.86***	**22.99***	**0.89**	**0.81**

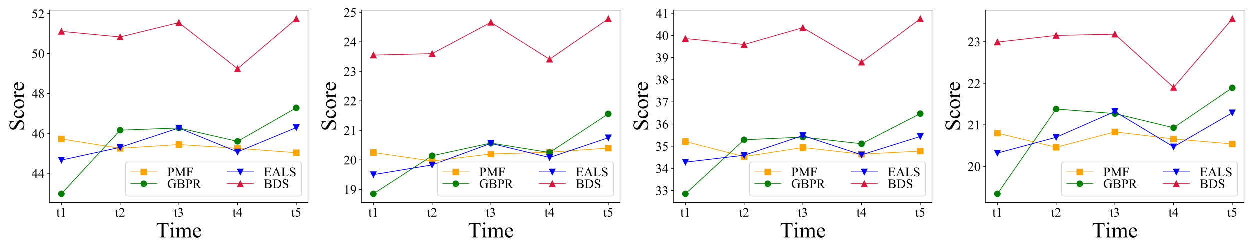
Table 5.1: Automatic & human evaluation results of my model compared with baselines. Symbol '*' highlights the cases where my model significantly beats all baselines with p value smaller than 0.01.

baseline on each automatic metric for all the ten runs, and apply a t-test on the ten differences to check whether the performance difference is significant.

RT performs the worst as the product title contains limited and static information to reveal product sentiment dynamics. From TR results, adding sparse reviews can improve model performance, but is still worse than the rest approaches, which strongly indicates the effectiveness of using neighbor product reviews for generating aspect summarization. Most of neighbor selection based baselines have roughly similar results except TS model, meaning that behavior based is better than content based neighbor selection. Surprisingly, Random model can achieve a relatively satisfying result. One possible reason is that because most of reviews in online shopping websites are positive, randomly sampled neighbor product might receive reviews containing relevant sentimental phrases. Howsoever, my BDS model outperforms all baselines significantly (*p<0.01*) on all metrics, demonstrating the superiority of my proposed reinforcement neighbor selection.

5.4.2 Human Evaluation

The right part of Table 5.1 reports the human evaluation result for all the models. Higher WTR and WCR scores indicate the related model can generate better structured summaries from human perspective. For each baseline, WTR and WCR scores are the pairwise comparison results with the BDS model. RT model performs the worst. And content based methods (RT and TR) also perform



(a) ROUGE-1 (b) ROUGE-2 (c) ROUGE-L (d) METEOR

Figure 5.2: Automatic evaluation results of my model compared with the Top 3 baselines in five consecutive time periods ($t_1 \sim t_5$).

worse than neighbor selection based methods in general. GBPR performs much better than the rest baselines. It beats my model in roughly 30% of summary pairs. The reported two scores of my model are the average of pairwise comparison results with all baselines, which shows that my model can beat other baselines on 90% of all summary pairs.

Both automatic and human evaluation results demonstrate content based baselines perform worse than neighbor selection based baselines. However, there are still some inconsistencies between their evaluation results. Although GBPR has similar performance with the rest of neighbor based baselines on automatic metrics, it unexpectedly outperforms on human metrics, which indicates how to organize sequences has huge impact on summary semantic quality.

5.4.3 Dynamic Performance

To better present my model dynamic performance, I visualize the automatic evaluation results of my model as well as the best three baselines (PMF, EALS and GBPR) in all five consecutive time periods, shown in Figure 5.2. The three best baselines are all neighbor selection approaches. Across all time periods, all four model performances are relatively consistent and follow similar trend. Their performances go down a little bit in the fourth time period but raise up immediately in the next time period. Among three baselines, GBPR achieves the best performance result over the rest two. It does not perform well in the beginning, but keeps growing and beats the rest baselines in later time periods. In general, the performances of three baselines are not far away from each other, especially from time period t_2 to t_4. Shown in Figure 5.2, their plotted lines are basically mingled

Model	RG-1	RG-2	RG-L	METEOR
– Behavior	-14.21	-12.63	-11.37	-5.72
– Category	-4.74	-2.94	-4.27	-1.65
– Profile	-5.78	-3.77	-5.33	-2.34

Table 5.2: Performance differences between the model with truncated input and original BDS model. '–' means removing related input component from my model.

together. However, my model achieves a far better evaluation result than baselines. Its plotted lines (red lines) are always significantly above the rest three lines in terms of all four evaluation metrics. Moreover, model performances on the four reported metrics are with similar trend. And the three ROUGE based metrics are with an even more similar trend than METEOR.

5.4.4 Input Component Evaluation

As aforementioned, my model requires three types of product input information: user behavior, product category, and profile descriptive phrases. To examine whether all involved information are effective, I conduct an extensive study by removing each type of information iteratively while holding the rest information fixed. Table 5.2 shows the performance difference between the models with truncated input and original BDS model. In detail, removing user behavior (product category) means that only product category (user behavior) is used for sentiment prediction pretraining. Removing product profiles means that only neighbor product sentimental phrases contribute to summarization generation.

From Table 5.2, removing related input component always leads a decrease on model performance, which indicates that all the three types of input component are useful in BDS model. Compared with product category and descriptive phrases, user behavior obviously has the most influential impact because removing it causes the largest drop in model performance over all four reported metrics. Surprisingly, it declines the model performance down close to the worst baseline, RT. It explicitly verifies the importance of user behavior for product summarization generation. Moreover, the performance decrease by removing profile descriptive phrases is larger than removing product category embedding. The reason might be that profile descriptive phrases are

Time	Ground Truth	BDS
t_1	The **material** of the dress is super **nice** and **soft to wear**. It is made of cotton and touches like a **high-end** dress. A perfect gift for aged women.	The dress has **good material** . It is made of **cotton and touches very soft**. A perfect gift for mums who have **high-standard** requirement on dress material.
t_2	The dress **material** is **not as good as promised**. It is **not good** for aged women with high requirements. It touches as **cheap material** and smells weird.	The **material is terrible**. It touches like **carded yarn with much cheaper material** as promised for mums. It has a weird smell when wearing it.

Table 5.3: A real case to show my model's generated dynamic product summarization for a dress on product material. The green color indicates positive words. While red color indicates negative words.

directly used for summarization generation. While product category contributes to the sentiment prediction pretraining, which only has implicit impact and is not able to directly reflect dynamic aspect sentiment changes.

5.4.5 Case Study

I conduct a case study in Table 5.3 on an actual dress to show how my model summarize its product summary changes in a timely manner. In this case, I only care about characteristic changes on product material and demonstrate material-related content from the full generated summaries. In the real world, the characteristic of the dress material goes from positive to negative (concluded from summarization) due to its manufacturer's counterfeit (reported by customers in time t_2). My model is able to detect this characteristic change from its updated customer shopping behaviors and dynamically locate neighbor products with similar issues (like material problems). From the result shown in Table 5.3, the generated summaries related to 'Material' supports my model effectiveness.

CHAPTER 6
CONCLUSION

6.1 Contributions

In this dissertation, I present an overview of community detection problems and three related research problems to explain how to solve community detection tasks and apply community information in other domains. Chapter 2 summarizes related works from various perspectives. Chapter 3 to Chapter 5 provide details on the three proposed research questions. In this Chapter, I summarize the contribution of each chapter in my dissertation as follows.

Chapter 2 is the literature review discusses the most outstanding community detection researches, and summarizes more than twice the number of research studies noted in previously published literature. To my knowledge, Chapter 2 is the most comprehensive literature review in the community detection domain. In this chapter, the selected papers were chosen from more than one thousand candidates are all published in the recent decade between 2010 and early 2020, which means they are timely, state-of-the-arts, and outstanding works. The selection criterion of paper was that a study must have been published in a top-tier journal or conference in its related domain or include at least 50 citations. I set 2020 as the time threshold to filter papers for two reasons: First, because several survey studies on community detection were published in the past five to ten years, I would duplicate my work if I paid too more attention to those classic works. Second, even though community detection is only a sub topic in graph mining, it has gained exponentially increased attention in recent years. Many studies are published in computer science, physics, and statistics domains. It is necessary to summarize such works so that future researchers can have a better understanding and global view of the community detection domain.

Based on ideas from previous researches [3, 4], the literature reviewed in Chapter 2 includes five categories. First, research to deal with different types of graphs is introduced. The graph types considered in this dissertation include heterogeneous and multilayer graphs, sparse graphs, dynamic graphs, large graphs and attribute graphs. The taxonomy covers most of the graph types that have appeared in community detection research. Second, different community detection tasks are also illustrated. As community detection is a complex domain, there are actually many works focusing on all kinds of trivial and subtle tasks. Among these numerous tasks, I selected five of the most interesting tasks and introduced ten to twenty works for each task: overlapping community detection task aims to solve the typical scenario where a node can belong to multiple communities; number of communities task argues how to explore and choose the best number of communities for many existing approaches; community search task is a type of partial community detection mixed with information retrieval; and enhanced and semi-supervised community detection are two other tasks widely discussed in current research studies. I also discuss how to apply community detection to other domains and how to evaluate and compare model performances in the dissertation. In general, these five mentioned categories cover most topics in community detection, which I trust I presented with clarity and preciseness in an easy-to-follow way.

In Chapter 3, I introduce a personalized community detection approach, which is the first attempt to address the topic of detecting communities with different resolutions in an attempt to meet user need. To solve this task, I proposed a model with an offline binary community tree construction step and an online genetic pruning step. A distributed version of the model is also deployed to accelerate running efficiency. Extensive experiments on two different datasets show that my proposed model outperforms all baselines in terms of accuracy and efficiency.

In Chapter 4, I address an outgrowth of the star-shaped topology of the current internet ecosystem. For example, giant service providers, such as Facebook, Google, and Amazon provide easy login channels to thousands of sites and apps, which means that users no longer need to undergo laborious account creation processes. This situation inspired me to propose a novel cross-graph community detection inquiry in order to detect user communities in sparse graphs by leveraging cross-graph information and pairwise learning techniques. This knowledge can substantially help small businesses and services identify and understand user groups and their interests. In the proposed model, a two-level filtering module helps to reduce the negative propagation effect from noisy and heterogeneous graphs. A well-defined Community Recurrent Unit (CRU) detects user community affiliations to support pairwise community closeness prediction. Extensive experiments on two real datasets demonstrate the superiority of the model compared with all baselines.

In Chapter 5, by leveraging communities detected from multi-type user behaviors instead of sparse reviews, I propose a multi-task model to solve an innovative dynamic product summarization task. Extensive experiments show the model is consistently promising and significantly outperforms the baselines. Being the first study on this newly proposed task, I aim to explore the relationship between user community and product summarization so as to address the cold-start problem, that it, products without recent reviews. As a result, the proposed model can provide a much larger scope in the e-commerce ecosystem while enabling explainable dynamic analysis on products. As the generated summarization is sensitive to customers, I would never wish to generate 'fake reviews' to mislead them. Instead, the summarization should be provided to online sellers only as auxiliary information.

6.2 Limitation

There are still some apparent limitations in the graph mining / community detection domain, the existing research environment, and the three projects introduced in this dissertation.

First, from a personal perspective, I firmly believe that graph mining needs be at the same level as natural language processing (NLP), computer vision (CV), and audio, as these four domains depict user profiles from four different perspectives: interaction, text, image, and audio. However, compared with the other three domains, graph mining suffers from a lack of attention In general. From an academic viewpoint, the other three domains have their particular top-tier journal and annual conference, such as ACL, CVPR, and ICASSP; whereas for graph mining or network analysis, there are as yet no competitive conferences (NetSci is not a paper-driven conference). This reality leads to fewer chances for researchers in the complex network domain to exchange ideas and undertake collaborations. Meanwhile, at the university level, there are fewer institutes and research groups in graph mining, compared with the other three domains. Among the few groups, Indiana University Center for Complex Networks and Systems Research and Stanford University's Jure Leskovec group are the two most outstanding research units. From an industry viewpoint, there is still a long way to go before applying graph mining techniques in real-world scenarios. To date, most community detection research is still more theoretical than applicable, residing in the physics, statistics, and mathematics domains. Even though I can see more and more real applications involving graph mining and even community detection techniques, these techniques are used mostly as supportive features embedded in other application pipelines.

I propose the development of more independent applications solely use graph mining techniques. To my knowledge, Amazon and Microsoft have their own teams focusing on knowledge graphs such as Deep Graph Library[1]. Alibaba also releases its open-source deep learning based graph mining framework, Euler[2], and the Graph Neural Network Platform, graph-learn[3]. This is a good sign that high-tech companies are starting to pay more attention to the graph mining domain.

Second, from a domain perspective, existing community detection researchers still face a lot of challenges. [284, 285, 286] explains the resolution limit of modularity maximization. [287] shows the limitations of spectral methods with regard to detectability threshold and localization of eigenvectors. Because of the No Free Lunch Theorem [1], no single optimal method can satisfy all types of community detection tasks. To evaluate model performance, more reliable benchmark graphs and evaluation metrics need to be explored, proposed and constructed. Moreover, scalability is another issue. Many tasks have proved to be NP hard problems, which means they are not able to be solved in large-scale graphs. How to efficiently approximate optimal solutions and how to run possible solutions in parallel are the two main potential tracks to explore.

Third, from the dissertation perspective, in the literature review chapter, the selected works were filtered from a pool of over one thousand candidates. However, even though these works are the most representative ones, more than 70% of high-quality works were filtered out. Moreover,

[1]https://www.dgl.ai/
[2]https://github.com/alibaba/euler
[3]https://github.com/alibaba/graph-learn

the literature review is a brief summarization of several of the most popular topics and ignore other smaller topics such as multi-view graphs and hierarchical community detection. Therefore, although Chapter 2 gives considerable insight about the research that has occurred in the latest decade, not all topics are covered in detail.

There are also limitations with respect to each of the three research tasks presented here. In the first personalized community detection task, the proposed model partially relies on existing models such as Infomap and Node2vec to construct the offline community tree and calculate pairwise node similarity. In the second cross-graph community detection task, In the cross-graph community detection task, it was necessary to empirically detect communities to generate pseudo-labels for training in a separate step, which consumes time and lacks reliability. In the third community-aware product summarization task, community is leveraged as a strong support to generate product summarizations. In the proposed model, both product behavior representation and behavior-to-community pretraining need to be determined apart from the multi-task model. Besides, as the proposed task is quite new, there are no baseline models to compare it with. I have to compromise and apply several constructed step-by-step models instead of end-to-end models directly proposed for this task.

6.3 Future Work

Looking ahead, my next steps are to keep track of state-of-the-art activities in the community detection domain. With respect to the personalized community detection task, as mentioned above, I plan to design my own method for offline community tree construction and node representation, preferably by having the entire offline process developed end-to-end without arbitrary parameters and methods selection. For the cross-graph community detection task, to avoid an extra step to detect communities, I want to launch the community detection process into the main pipeline to achieve a real end-to-end model, as well as include more textual information from users in the model in order to construct attribute graphs. For the community-aware dynamic product summarization, a further step will be to integrate community detection into the joint training process. In addition more advanced text generative models need be utilized, such as Pointer Network [288] and Copynet [289].

On a boarder note, more conferences on community detection are needed to offer researchers the opportunity to unite, exchange ideas and collaborate, .and make themselves known to researchers in other domains. Moreover, from a methodological perspective, deep learning should be a future trend in community detection, as graph neural networks (GNN) are already the main trend in graph mining. Future work can either transfer existing models such as modularity-based or statistical inference-based (stochastic block model) models to their deep learning version or develop new models via GNN to incorporate complex graphs (i.e., large-scale graphs, dynamic graphs, and heterogeneous graphs). From an application perspective, better open-source frameworks need be developed and shared with researchers so that they can easily deploy and run models on those frameworks. Meanwhile, researchers should be attentive in exploring other scenarios where community detection techniques can be plugged in as extra support.

REFERENCES

[1] David H Wolpert and William G Macready. “No free lunch theorems for optimization”. In: *IEEE transactions on evolutionary computation* 1.1 (1997), pp. 67–82.

[2] Mark EJ Newman. “Communities, modules and large-scale structure in networks”. In: *Nature physics* 8.1 (2012), pp. 25–31.

[3] Santo Fortunato. “Community detection in graphs”. In: *Physics reports* 486.3-5 (2010), pp. 75–174.

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

[5] Michele Coscia, Fosca Giannotti, and Dino Pedreschi. “A classification for community discovery methods in complex networks”. In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 4.5 (2011), pp. 512–546.

[6] Fragkiskos D Malliaros and Michalis Vazirgiannis. “Clustering and community detection in directed networks: A survey”. In: *Physics Reports* 533.4 (2013), pp. 95–142.

[7] Steve Harenberg et al. “Community detection in large-scale networks: a survey and empirical evaluation”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 6.6 (2014), pp. 426–439.

[8] Jungeun Kim and Jae-Gil Lee. “Community detection in multi-layer graphs: A survey”. In: *ACM SIGMOD Record* 44.3 (2015), pp. 37–48.

[9] Emmanuel Abbe. “Community detection and stochastic block models: recent developments”. In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 6446–6531.

[10] Jierui Xie, Stephen Kelley, and Boleslaw K Szymanski. “Overlapping community detection in networks: The state-of-the-art and comparative study”. In: *Acm computing surveys (csur)* 45.4 (2013), p. 43.

[11] Alessia Amelio and Clara Pizzuti. “Overlapping community discovery methods: a survey”. In: *Social Networks: Analysis and Case Studies*. Springer, 2014, pp. 105–125.

[12] Maria CV Nascimento and Andre CPLF De Carvalho. “Spectral methods for graph clustering–a survey”. In: *European Journal of Operational Research* 211.2 (2011), pp. 221–231.

[13] Muhammad Aqib Javed et al. “Community detection in networks: A multidisciplinary review”. In: *Journal of Network and Computer Applications* 108 (2018), pp. 87–111.

[14] Punam Bedi and Chhavi Sharma. “Community detection in social networks”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 6.3 (2016), pp. 115–135.

[15] Tanmoy Chakraborty et al. “Metrics for community analysis: A survey”. In: *ACM Computing Surveys (CSUR)* 50.4 (2017), p. 54.

[16] Aditya Grover and Jure Leskovec. “node2vec: Scalable feature learning for networks”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2016, pp. 855–864.

[17] Zheng Gao et al. “edge2vec: Representation learning using edge semantics for biomedical knowledge discovery”. In: *BMC bioinformatics* 20.1 (2019), p. 306.

[18] Yizhou Sun et al. “Pathselclus: Integrating meta-path selection with user-guided object clustering in heterogeneous information networks”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 7.3 (2013), p. 11.

[19] Manish Gupta, Jing Gao, and Jiawei Han. “Community distribution outlier detection in heterogeneous information networks”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2013, pp. 557–573.

[20] Yizhou Sun, Charu C Aggarwal, and Jiawei Han. “Relation strength-aware clustering of heterogeneous information networks with incomplete attributes”. In: *Proceedings of the VLDB Endowment* 5.5 (2012), pp. 394–405.

[21] Dongxiao He et al. “A Stochastic Model for Detecting Heterogeneous Link Communities in Complex Networks.” In: *AAAI*. 2015, pp. 130–136.

[22] Yang Zhou and Ling Liu. “Social influence based clustering of heterogeneous information networks”. In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2013, pp. 338–346.

[23] Mingqing Huang et al. “Overlapping community detection in heterogeneous social networks via the user model”. In: *Information Sciences* 432 (2018), pp. 164–184.

[24] Xiang Li et al. “Semi-supervised clustering in attributed heterogeneous information networks”. In: *Proceedings of the 26th International Conference on World Wide Web*. 2017, pp. 1621–1629.

[25] Marya Bazzi et al. “Community detection in temporal multilayer networks, with an application to correlation networks”. In: *Multiscale Modeling & Simulation* 14.1 (2016), pp. 1–41.

[26] Manlio De Domenico et al. “Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems”. In: *Physical Review X* 5.1 (2015), p. 11027.

[27] Toni Valles-Catala et al. “Multilayer stochastic block models reveal the multilayer structure of complex networks”. In: *Physical Review X* 6.1 (2016), p. 11036.

[28] Ling Huang, Chang-Dong Wang, and Hong-Yang Chao. “A harmonic motif modularity approach for multi-layer network community detection”. In: *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2018, pp. 1043–1048.

[29] Florent Krzakala et al. “Spectral redemption in clustering sparse networks”. In: *Proceedings of the National Academy of Sciences* 110.52 (2013), pp. 20935–20940.

[30] Yudong Chen, Sujay Sanghavi, and Huan Xu. “Clustering sparse graphs”. In: *Advances in neural information processing systems*. 2012, pp. 2204–2212.

[31] Arash A Amini et al. “Pseudo-likelihood methods for community detection in large sparse networks”. In: *The Annals of Statistics* 41.4 (2013), pp. 2097–2122.

[32] Atieh Mirshahvalad et al. “Significant communities in large sparse networks”. In: *PloS one* 7.3 (2012).

[33] Jess Banks et al. “Information-theoretic thresholds for community detection in sparse networks”. In: *Conference on Learning Theory*. 2016, pp. 383–416.

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

[35] Olivier Guédon and Roman Vershynin. “Community detection in sparse networks via Grothendieck’s inequality”. In: *Probability Theory and Related Fields* 165.3-4 (2016), pp. 1025–1049.

[36] Tanja Hartmann, Andrea Kappes, and Dorothea Wagner. “Clustering evolving networks”. In: *Algorithm Engineering*. Springer, 2016, pp. 280–329.

[37] Yudong Chen, Vikas Kawadia, and Rahul Urgaonkar. “Detecting overlapping temporal community structure in time-evolving networks”. In: *arXiv preprint arXiv:1303.7226* (2013).

[38] Francesco Folino and Clara Pizzuti. “An evolutionary multiobjective approach for community discovery in dynamic networks”. In: *IEEE Transactions on Knowledge and Data Engineering* 26.8 (2013), pp. 1838–1852.

[39] Kevin S Xu and Alfred O Hero. “Dynamic stochastic blockmodels for time-evolving social networks”. In: *IEEE Journal of Selected Topics in Signal Processing* 8.4 (2014), pp. 552–562.

[40] Kevin Xu. “Stochastic block transition models for dynamic networks”. In: *Artificial Intelligence and Statistics*. 2015, pp. 1079–1087.

[41] Chang-Dong Wang, Jian-Huang Lai, and S Yu Philip. “NEIWalk: community discovery in dynamic content-based networks”. In: *IEEE transactions on knowledge and data engineering* 26.7 (2013), pp. 1734–1748.

[42] Ana Paula Appel et al. “Temporally evolving community detection and prediction in content-centric networks”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2018, pp. 3–18.

[43] Chang-Dong Wang, Jian-Huang Lai, and Philip S Yu. “Dynamic community detection in weighted graph streams”. In: *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM. 2013, pp. 151–161.

[44] Myunghwan Kim and Jure Leskovec. “Nonparametric multi-group membership model for dynamic networks”. In: *Advances in neural information processing systems*. 2013, pp. 1385–1393.

[45] Amir Ghasemian et al. “Detectability thresholds and optimal algorithms for community structure in dynamic networks”. In: *Physical Review X* 6.3 (2016), p. 31005.

[46] Sikun Yang and Heinz Koeppl. “A poisson gamma probabilistic model for latent node-group memberships in dynamic networks”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[47] Tiago P Peixoto and Martin Rosvall. “Modelling sequences and temporal networks with dynamic community structures”. In: *Nature communications* 8.1 (2017), p. 582.

[48] J-C Delvenne, Sophia N Yaliraki, and Mauricio Barahona. “Stability of graph communities across time scales”. In: *Proceedings of the National Academy of Sciences* (2010).

[49] Laetitia Gauvin, André Panisson, and Ciro Cattuto. “Detecting the community structure and activity patterns of temporal networks: a non-negative tensor factorization approach”. In: *PloS one* 9.1 (2014).

[50] Lei Tang, Huan Liu, and Jianping Zhang. "Identifying evolving groups in dynamic multi-mode networks". In: *IEEE Transactions on Knowledge and Data Engineering* 24.1 (2011), pp. 72–85.

[51] Kathy Macropol and Ambuj Singh. "Scalable discovery of best clusters on large graphs". In: *Proceedings of the VLDB Endowment* 3.1-2 (2010), pp. 693–702.

[52] Daniel A Spielman and Shang-Hua Teng. "A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning". In: *SIAM Journal on computing* 42.1 (2013), pp. 1–26.

[53] Venu Satuluri, Srinivasan Parthasarathy, and Yiye Ruan. "Local graph sparsification for scalable clustering". In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. 2011, pp. 721–732.

[54] Liaoruo Wang et al. "Detecting community kernels in large social networks". In: *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE. 2011, pp. 784–793.

[55] Joyce Jiyoung Whang, Xin Sui, and Inderjit S Dhillon. "Scalable and memory-efficient clustering of large-scale social networks". In: *2012 IEEE 12th International Conference on Data Mining*. IEEE. 2012, pp. 705–714.

[56] Yakun Li et al. "Efficient community detection with additive constrains on large networks". In: *Knowledge-Based Systems* 52 (2013), pp. 268–278.

[57] Jialu Liu et al. "Large-scale spectral clustering on graphs". In: *Twenty-Third International Joint Conference on Artificial Intelligence*. 2013.

[58] Pasquale De Meo et al. "Mixing local and global information for community detection in large networks". In: *Journal of Computer and System Sciences* 80.1 (2014), pp. 72–87.

[59] George Kollios, Michalis Potamias, and Evimaria Terzi. "Clustering large probabilistic graphs". In: *IEEE Transactions on Knowledge and Data Engineering* 25.2 (2013), pp. 325–336.

[60] Arnau Prat-Pérez, David Dominguez-Sal, and Josep-Lluis Larriba-Pey. “High quality, scalable and parallel community detection for large real graphs”. In: *Proceedings of the 23rd international conference on World wide web*. 2014, pp. 225–236.

[61] Charalampos E Tsourakakis, Jakub Pachocki, and Michael Mitzenmacher. “Scalable motif-aware graph clustering”. In: *Proceedings of the 26th International Conference on World Wide Web*. 2017, pp. 1451–1460.

[62] Yixuan Li et al. “Uncovering the small community structure in large networks: A local spectral approach”. In: *Proceedings of the 24th international conference on world wide web*. 2015, pp. 658–668.

[63] Vandana Bhatia and Rinkle Rani. “Dfuzzy: a deep learning-based fuzzy clustering model for large graphs”. In: *Knowledge and Information Systems* 57.1 (2018), pp. 159–181.

[64] David Hallac, Jure Leskovec, and Stephen Boyd. “Network lasso: Clustering and optimization in large graphs”. In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM. 2015, pp. 387–396.

[65] Lucas GS Jeub et al. “Think locally, act locally: Detection of small, medium-sized, and large communities in large networks”. In: *Physical Review E* 91.1 (2015), p. 12821.

[66] Tiago P Peixoto. “Model selection and hypothesis testing for large-scale network models with overlapping groups”. In: *Physical Review X* 5.1 (2015), p. 11033.

[67] Jaewon Yang, Julian McAuley, and Jure Leskovec. “Community detection in networks with node attributes”. In: *Data Mining (ICDM), 2013 IEEE 13th international conference on*. IEEE. 2013, pp. 1151–1156.

[68] Zhiqiang Xu et al. “A model-based approach to attributed graph clustering”. In: *Proceedings of the 2012 ACM SIGMOD international conference on management of data*. 2012, pp. 505–516.

[69] Yu Han and Jie Tang. "Probabilistic community and role model for social networks". In: *Proceedings of the 21th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*. 2015, pp. 407–416.

[70] Aleksandar Bojchevski and Stephan Günnemann. "Bayesian robust attributed graph clustering: Joint learning of partial anomalies and group structure". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[71] Dongxiao He et al. "Joint identification of network communities and semantics via integrative modeling of network topologies and node contents". In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[72] Guo-Jun Qi, Charu C Aggarwal, and Thomas Huang. "Community detection with edge content in social media networks". In: *2012 IEEE 28th International Conference on Data Engineering*. IEEE. 2012, pp. 534–545.

[73] Xiao Wang et al. "Semantic community identification in large attribute networks". In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[74] Meng Qin et al. "Adaptive community detection incorporating topology and content in social networks". In: *Knowledge-Based Systems* 161 (2018), pp. 342–356.

[75] Xiaotong Zhang et al. "Attributed graph clustering via adaptive graph convolution". In: *arXiv preprint arXiv:1906.01210* (2019).

[76] Di Jin et al. "Graph convolutional networks meet Markov random fields: Semi-supervised community detection in attribute networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 152–159.

[77] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. "Clustering large attributed graphs: An efficient incremental approach". In: *2010 IEEE International Conference on Data Mining*. IEEE. 2010, pp. 689–698.

[78] Simon Pool, Francesco Bonchi, and Matthijs van Leeuwen. “Description-driven community detection”. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 5.2 (2014), pp. 1–28.

[79] Xin Huang, Hong Cheng, and Jeffrey Xu Yu. “Dense community detection in multi-valued attributed networks”. In: *Information Sciences* 314 (2015), pp. 77–99.

[80] Jaewon Yang and Jure Leskovec. “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. 2013, pp. 587–596.

[81] Michele Coscia et al. “Demon: a local-first discovery method for overlapping communities”. In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2012, pp. 615–623.

[82] Joyce Jiyoung Whang, David F Gleich, and Inderjit S Dhillon. “Overlapping community detection using seed set expansion”. In: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2013, pp. 2099–2108.

[83] Joyce Jiyoung Whang, David F Gleich, and Inderjit S Dhillon. “Overlapping community detection using neighborhood-inflated seed expansion”. In: *IEEE Transactions on Knowledge and Data Engineering* 28.5 (2016), pp. 1272–1284.

[84] Xiaofeng Wang, Gongshen Liu, and Jianhua Li. “Overlapping Community Detection Based on Structural Centrality in Complex Networks”. In: *IEEE Access* 5 (2017), pp. 25258–25269.

[85] Hongyi Zhang, Irwin King, and Michael R Lyu. “Incorporating Implicit Link Preference Into Overlapping Community Detection.” In: *AAAI*. 2015, pp. 396–402.

[86] Hongyi Zhang et al. “Modeling the Homophily Effect between Links and Communities for Overlapping Community Detection.” In: *IJCAI*. 2016, pp. 3938–3944.

[87] Justine Eustace, Xingyuan Wang, and Yaozu Cui. “Overlapping community detection using neighborhood ratio matrix”. In: *Physica A: Statistical Mechanics and its Applications* 421 (2015), pp. 510–521.

[88] Di Jin, Bogdan Gabrys, and Jianwu Dang. “Combined node and link partitions method for finding overlapping communities in complex networks”. In: *Scientific reports* 5 (2015), p. 8600.

[89] Prem K Gopalan and David M Blei. “Efficient discovery of overlapping communities in massive networks”. In: *Proceedings of the National Academy of Sciences* 110.36 (2013), pp. 14534–14539.

[90] Di Jin et al. “Detect overlapping communities via ranking node popularities”. In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[91] Jaewon Yang and Jure Leskovec. “Community-affiliation graph model for overlapping network community detection”. In: *2012 IEEE 12th international conference on data mining*. IEEE. 2012, pp. 1170–1175.

[92] Kun He et al. “Detecting overlapping communities from local spectral subspaces”. In: *2015 IEEE International Conference on Data Mining*. IEEE. 2015, pp. 769–774.

[93] Jin-Xuan Yang and Xiao-Dong Zhang. “Finding overlapping communities using seed set”. In: *Physica A: Statistical Mechanics and its Applications* 467 (2017), pp. 96–106.

[94] Mark EJ Newman and Gesine Reinert. “Estimating the number of communities in a network”. In: *Physical review letters* 117.7 (2016), p. 78301.

[95] Can M Le and Elizaveta Levina. “Estimating the number of communities in networks by spectral methods”. In: *arXiv preprint arXiv:1507.00827* (2015).

[96] D Franco Saldana, Yi Yu, and Yang Feng. “How many communities are there?” In: *Journal of Computational and Graphical Statistics* 26.1 (2017), pp. 171–181.

[97] Tatsuro Kawamoto and Yoshiyuki Kabashima. “Cross-validation estimate of the number of clusters in a network”. In: *Scientific reports* 7.1 (2017), pp. 1–17.

[98] Kehui Chen and Jing Lei. “Network cross-validation for determining the number of communities in network data”. In: *Journal of the American Statistical Association* 113.521 (2018), pp. 241–251.

[99] Tatsuro Kawamoto and Yoshiyuki Kabashima. “Comparative analysis on the selection of number of clusters in community detection”. In: *Physical Review E* 97.2 (2018), p. 022315.

[100] Mauro Sozio and Aristides Gionis. “The community-search problem and how to plan a successful cocktail party”. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010, pp. 939–948.

[101] Wanyun Cui et al. “Local search of communities in large graphs”. In: *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM. 2014, pp. 991–1002.

[102] Nicola Barbieri et al. “Efficient and effective community search”. In: *Data mining and knowledge discovery* 29.5 (2015), pp. 1406–1433.

[103] Yubao Wu et al. “Robust local community detection: on free rider effect and its elimination”. In: *Proceedings of the VLDB Endowment* 8.7 (2015), pp. 798–809.

[104] Xin Huang et al. “Approximate closest community search in networks”. In: *arXiv preprint arXiv:1505.05956* (2015).

[105] Jie Chen and Yousef Saad. “Dense subgraph extraction with application to community detection”. In: *IEEE Transactions on Knowledge and Data Engineering* 24.7 (2012), pp. 1216–1230.

[106] Lu Qin et al. “Locally densest subgraph discovery”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2015, pp. 965–974.

[107] Andrea Lancichinetti et al. “Finding statistically significant communities in networks”. In: *PloS one* 6.4 (2011), e18961.

[108] Xin Huang et al. “Querying k-truss community in large and dynamic graphs”. In: *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 2014, pp. 1311–1322.

[109] Zibin Zheng et al. “Finding weighted k-truss communities in large networks”. In: *Information Sciences* 417 (2017), pp. 344–360.

[110] Yaowei Yan et al. “Constrained local graph clustering by colored random walk”. In: *The world wide web conference*. 2019, pp. 2137–2146.

[111] Rong-Hua Li et al. “Influential community search in large networks”. In: *Proceedings of the VLDB Endowment* 8.5 (2015), pp. 509–520.

[112] Alireza Khadivi, Ali Ajdari Rad, and Martin Hasler. “Network community-detection enhancement by proper weighting”. In: *Physical Review E* 83.4 (2011), p. 46104.

[113] Pasquale De Meo et al. “Enhancing community detection using a network weighting strategy”. In: *Information Sciences* 222 (2013), pp. 648–668.

[114] Darong Lai, Hongtao Lu, and Christine Nardini. “Enhanced modularity-based community detection by random walk network preprocessing”. In: *Physical Review E* 81.6 (2010), p. 66118.

[115] Haoran Wen, EA Leicht, and Raissa M D’Souza. “Improving community detection in networks by targeted node removal”. In: *Physical Review E* 83.1 (2011), p. 16114.

[116] Dongxiao He et al. “A model framework for the enhancement of community detection in complex networks”. In: *Physica A: Statistical Mechanics and its Applications* 461 (2016), pp. 602–612.

[117] Zhong-Yuan Zhang, Kai-Di Sun, and Si-Qi Wang. “Enhanced community structure detection in complex networks with partial background information”. In: *Scientific reports* 3 (2013), p. 3241.

[118] Jiajun Zhou et al. “Adversarial Enhancement for Community Detection in Complex Networks”. In: *arXiv preprint arXiv:1911.01670* (2019).

[119] Pei-Zhen Li et al. “EdMot: An Edge Enhancement Approach for Motif-aware Community Detection”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 479–487.

[120] Nate Veldt, David Gleich, and Anthony Wirth. “Learning resolution parameters for graph clustering”. In: *The World Wide Web Conference*. 2019, pp. 1909–1919.

[121] Shreyansh Bhatt et al. “Knowledge graph enhanced community detection and characterization”. In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 2019, pp. 51–59.

[122] Xiaoke Ma et al. “Semi-supervised clustering algorithm for community structure detection in complex networks”. In: *Physica A: Statistical Mechanics and its Applications* 389.1 (2010), pp. 187–197.

[123] Xiao Liu et al. “Semi-supervised community detection based on non-negative matrix factorization with node popularity”. In: *Information Sciences* 381 (2017), pp. 304–321.

[124] Lei Li et al. “Extremal optimization-based semi-supervised algorithm with conflict pairwise constraints for community detection”. In: *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*. IEEE. 2014, pp. 180–187.

[125] Jianjun Cheng et al. “Active semi-supervised community detection based on must-link and cannot-link constraints”. In: *PloS one* 9.10 (2014).

[126] Liang Yang et al. “Active link selection for efficient semi-supervised community detection”. In: *Scientific reports* 5 (2015), p. 9039.

[127] Wenjun Wang et al. “A Unified Weakly Supervised Framework for Community Detection and Semantic Matching”. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2018, pp. 218–230.

[128] Eric Eaton and Rachael Mansbach. “A spin-glass model for semi-supervised community detection”. In: *Twenty-Sixth AAAI Conference on Artificial Intelligence*. 2012.

[129] Pan Zhang, Cristopher Moore, and Lenka Zdeborová. “Phase transitions in semisupervised clustering of sparse networks”. In: *Physical Review E* 90.5 (2014), p. 52802.

[130] Liang Yang et al. “A unified semi-supervised community detection framework using latent space graph regularization”. In: *IEEE transactions on cybernetics* 45.11 (2014), pp. 2585–2598.

[131] Mark EJ Newman. “Fast algorithm for detecting community structure in networks”. In: *Physical review E* 69.6 (2004), p. 66133.

[132] Mark EJ Newman. “Modularity and community structure in networks”. In: *Proceedings of the national academy of sciences* 103.23 (2006), pp. 8577–8582.

[133] Vincent D Blondel et al. “Fast unfolding of communities in large networks”. In: *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008.

[134] Vincent A Traag, Ludo Waltman, and Nees Jan van Eck. “From Louvain to Leiden: guaranteeing well-connected communities”. In: *Scientific reports* 9.1 (2019), pp. 1–12.

[135] Zhan Bu et al. “A fast parallel modularity optimization algorithm (FPMQA) for community detection in online social network”. In: *Knowledge-Based Systems* 50 (2013), pp. 246–259.

[136] Jonathan Q Jiang and Lisa J McQuay. “Modularity functions maximization with nonnegative relaxation facilitates community detection in networks”. In: *Physica A: Statistical Mechanics and its Applications* 391.3 (2012), pp. 854–865.

[137] Vincenzo Nicosia et al. “Extending the definition of modularity to directed graphs with overlapping communities”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2009.3 (2009), P03024.

[138] Sonia Cafieri, Pierre Hansen, and Leo Liberti. “Locally optimal heuristic for modularity maximization of networks”. In: *Physical Review E* 83.5 (2011), p. 56105.

[139] Ju Xiang et al. “Local modularity for community detection in complex networks”. In: *Physica A: Statistical Mechanics and its Applications* 443 (2016), pp. 451–459.

[140] Liang Yang et al. “Modularity Based Community Detection with Deep Learning.” In: *IJCAI*. Vol. 16. 2016, pp. 2252–2258.

[141] Mingming Chen, Konstantin Kuzmin, and Boleslaw K Szymanski. “Community detection via maximization of modularity and its variants”. In: *IEEE Transactions on Computational Social Systems* 1.1 (2014), pp. 46–65.

[142] Peng Gang Sun, Lin Gao, and Yang Yang. “Maximizing modularity intensity for community partition and evolution”. In: *Information Sciences* 236 (2013), pp. 83–92.

[143] James P Bagrow. “Communities and bottlenecks: Trees and treelike networks have high modularity”. In: *Physical Review E* 85.6 (2012), p. 66118.

[144] Shuqin Zhang and Hongyu Zhao. “Normalized modularity optimization method for community identification with degree adjustment”. In: *Physical Review E* 88.5 (2013), p. 52802.

[145] Mark EJ Newman. “Equivalence between modularity optimization and maximum likelihood methods for community detection”. In: *Physical Review E* 94.5 (2016), p. 52315.

[146] Ling Chen, Qiang Yu, and Bolun Chen. “Anti-modularity and anti-community detecting in complex networks”. In: *Information Sciences* 275 (2014), pp. 293–313.

[147] Jing Lei, Alessandro Rinaldo, et al. “Consistency of spectral clustering in stochastic block models”. In: *The Annals of Statistics* 43.1 (2015), pp. 215–237.

[148] Mark EJ Newman. “Spectral methods for community detection and graph partitioning”. In: *Physical Review E* 88.4 (2013), p. 42822.

[149] Joan Bruna et al. “Spectral networks and locally connected networks on graphs”. In: *arXiv preprint arXiv:1312.6203* (2013).

[150] Alaa Saade, Florent Krzakala, and Lenka Zdeborová. “Spectral clustering of graphs with the bethe hessian”. In: *Advances in Neural Information Processing Systems*. 2014, pp. 406–414.

[151] Raj Rao Nadakuditi and Mark EJ Newman. “Graph spectra and the detectability of community structure in networks”. In: *Physical review letters* 108.18 (2012), p. 188701.

[152] Karl Rohe, Sourav Chatterjee, Bin Yu, et al. “Spectral clustering and the high-dimensional stochastic blockmodel”. In: *The Annals of Statistics* 39.4 (2011), pp. 1878–1915.

[153] Kamalika Chaudhuri, Fan Chung, and Alexander Tsiatas. “Spectral clustering of graphs with general degrees in the extended planted partition model”. In: *Conference on Learning Theory*. 2012, pp. 35–1.

[154] Michael W Mahoney, Lorenzo Orecchia, and Nisheeth K Vishnoi. “A local spectral method for graphs: With applications to improving graph partitions and exploring data graphs locally”. In: *Journal of Machine Learning Research* 13.Aug (2012), pp. 2339–2365.

[155] Xiao Zhang and Mark EJ Newman. “Multiway spectral community detection in networks”. In: *Physical Review E* 92.5 (2015), p. 52808.

[156] Antony Joseph, Bin Yu, et al. “Impact of regularization on spectral clustering”. In: *The Annals of Statistics* 44.4 (2016), pp. 1765–1791.

[157] Jiashun Jin et al. “Fast community detection by SCORE”. In: *The Annals of Statistics* 43.1 (2015), pp. 57–89.

[158] Austin R Benson, David F Gleich, and Jure Leskovec. “Tensor spectral clustering for partitioning higher-order network structures”. In: *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM. 2015, pp. 118–126.

[159] Xiang Li et al. “Spectral Clustering in Heterogeneous Information Networks”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 4221–4228.

[160] Pedro Mercado, Francesco Tudisco, and Matthias Hein. “Spectral Clustering of Signed Graphs via Matrix Power Means”. In: *International Conference on Machine Learning*. 2019, pp. 4526–4536.

[161] Lingfei Wu et al. “Scalable spectral clustering using random binning features”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018, pp. 2506–2515.

[162] Yilin Zhang and Karl Rohe. “Understanding regularized spectral clustering via graph conductance”. In: *Advances in Neural Information Processing Systems*. 2018, pp. 10631–10640.

[163] Brian Karrer and Mark EJ Newman. “Stochastic blockmodels and community structure in networks”. In: *Physical review E* 83.1 (2011), p. 16107.

[164] Elchanan Mossel, Joe Neeman, and Allan Sly. “Belief propagation, robust reconstruction and optimal recovery of block models”. In: *Conference on Learning Theory*. 2014, pp. 356–370.

[165] Elchanan Mossel and Jiaming Xu. “Density evolution in the degree-correlated stochastic block model”. In: *Conference on Learning Theory*. 2016, pp. 1319–1356.

[166] Tiago P Peixoto. “Entropy of stochastic blockmodel ensembles”. In: *Physical Review E* 85.5 (2012), p. 56122.

[167] Pierre Latouche, Etienne Birmelé, Christophe Ambroise, et al. “Overlapping stochastic block models with application to the french political blogosphere”. In: *The Annals of Applied Statistics* 5.1 (2011), pp. 309–336.

[168] Emmanuel Abbe and Colin Sandon. “Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery”. In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE. 2015, pp. 670–688.

[169] Tai Qin and Karl Rohe. “Regularized spectral clustering under the degree-corrected stochastic blockmodel”. In: *Advances in Neural Information Processing Systems*. 2013, pp. 3120–3128.

[170] Anderson Y Zhang, Harrison H Zhou, et al. “Minimax rates of community detection in stochastic block models”. In: *The Annals of Statistics* 44.5 (2016), pp. 2252–2280.

[171] Yunpeng Zhao, Elizaveta Levina, Ji Zhu, et al. “Consistency of community detection in networks under degree-corrected stochastic block models”. In: *The Annals of Statistics* 40.4 (2012), pp. 2266–2292.

[172] Alain Celisse, Jean-Jacques Daudin, Laurent Pierre, et al. “Consistency of maximum-likelihood and variational estimators in the stochastic block model”. In: *Electronic Journal of Statistics* 6 (2012), pp. 1847–1899.

[173] Xiaoran Yan et al. “Model selection for degree-corrected block models”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2014.5 (2014), P05007.

[174] Se-Young Yun and Alexandre Proutiere. “Optimal cluster recovery in the labeled stochastic block model”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 965–973.

[175] Tiago P Peixoto. “Efficient Monte Carlo and greedy heuristic for the inference of stochastic block models”. In: *Physical Review E* 89.1 (2014), p. 12804.

[176] Jiaming Xu, Laurent Massoulié, and Marc Lelarge. “Edge label inference in generalized stochastic block models: from spectral theory to impossibility results”. In: *Conference on Learning Theory*. 2014, pp. 903–920.

[177] YX Rachel Wang, Peter J Bickel, et al. “Likelihood-based model selection for stochastic block models”. In: *The Annals of Statistics* 45.2 (2017), pp. 500–528.

[178] Jing Lei et al. “A goodness-of-fit test for stochastic block models”. In: *The Annals of Statistics* 44.1 (2016), pp. 401–424.

[179] Chao Gao et al. “Community detection in degree-corrected block models”. In: *The Annals of Statistics* 46.5 (2018), pp. 2153–2185.

[180] Peihao Huang et al. “Deep embedding network for clustering”. In: *2014 22nd International conference on pattern recognition*. IEEE. 2014, pp. 1532–1537.

[181] Fei Tian et al. “Learning deep representations for graph clustering”. In: *Twenty-Eighth AAAI Conference on Artificial Intelligence*. 2014.

[182] Bing-Jie Sun et al. “A Non-negative Symmetric Encoder-Decoder Approach for Community Detection”. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM. 2017, pp. 597–606.

[183] Chun Wang et al. “Mgae: Marginalized graph autoencoder for graph clustering”. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2017, pp. 889–898.

[184] Sandro Cavallari et al. “Learning community embedding with community detection and node embedding on graphs”. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM. 2017, pp. 377–386.

[185] Fan-Yun Sun et al. “vGraph: A Generative Model for Joint Community Detection and Node Representation Learning”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 512–522.

[186] Ming Shao et al. “Deep Linear Coding for Fast Graph Clustering.” In: *IJCAI*. 2015, pp. 3798–3804.

[187] Vincent W Zheng et al. “From node embedding to community embedding”. In: *arXiv preprint arXiv:1610.09950* (2016).

[188] Jian Tang et al. “Line: Large-scale information network embedding”. In: *Proceedings of the 24th international conference on world wide web*. 2015, pp. 1067–1077.

[189] Benedek Rozemberczki et al. “Gemsec: Graph embedding with self clustering”. In: *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 2019, pp. 65–72.

[190] Joan Bruna and X Li. “Community detection with graph neural networks”. In: *Stat* 1050 (2017), p. 27.

[191] Wei-Lin Chiang et al. “Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 257–266.

[192] Dongsheng Luo et al. “Deep Multi-Graph Clustering via Attentive Cross-Graph Association”. In: *Proceedings of the 13th International Conference on Web Search and Data Mining*. 2020, pp. 393–401.

[193] Yuting Jia et al. “Communitygan: Community detection with generative adversarial nets”. In: *The World Wide Web Conference*. 2019, pp. 784–794.

[194] Fei Wang et al. “Community discovery using nonnegative matrix factorization”. In: *Data Mining and Knowledge Discovery* 22.3 (2011), pp. 493–521.

[195] Da Kuang, Chris Ding, and Haesun Park. “Symmetric nonnegative matrix factorization for graph clustering”. In: *Proceedings of the 2012 SIAM international conference on data mining*. SIAM. 2012, pp. 106–117.

[196] Da Kuang, Sangwoon Yun, and Haesun Park. “SymNMF: nonnegative low-rank approximation of a similarity matrix for graph clustering”. In: *Journal of Global Optimization* 62.3 (2015), pp. 545–574.

[197] Zhirong Yang et al. “Clustering by nonnegative matrix factorization using graph random walk”. In: *Advances in Neural Information Processing Systems*. 2012, pp. 1079–1087.

[198] Xianchao Tang et al. "Uncovering community structures with initialized bayesian nonnegative matrix factorization". In: *PloS one* 9.9 (2014).

[199] In: ().

[200] Yu Zhang and Dit-Yan Yeung. "Overlapping community detection via bounded nonnegative matrix tri-factorization". In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2012, pp. 606–614.

[201] Zhong-Yuan Zhang, Yong Wang, and Yong-Yeol Ahn. "Overlapping community detection in complex networks using symmetric binary matrix factorization". In: *Physical Review E* 87.6 (2013), p. 62803.

[202] Xiaohua Shi et al. "Community detection in social network with pairwisely constrained symmetric non-negative matrix factorization". In: *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*. 2015, pp. 541–546.

[203] Fanhua Shang, LC Jiao, and Fei Wang. "Graph dual regularization non-negative matrix factorization for co-clustering". In: *Pattern Recognition* 45.6 (2012), pp. 2237–2250.

[204] Yulong Pei, Nilanjan Chakraborty, and Katia Sycara. "Nonnegative matrix tri-factorization with graph regularization for community detection in social networks". In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*. 2015.

[205] Wenhui Wu et al. "Nonnegative matrix factorization with mixed hypergraph regularization for community detection". In: *Information Sciences* 435 (2018), pp. 263–281.

[206] Shudong Huang et al. "Robust graph regularized nonnegative matrix factorization for clustering". In: *Data Mining and Knowledge Discovery* 32.2 (2018), pp. 483–503.

[207] Fanghua Ye, Chuan Chen, and Zibin Zheng. "Deep autoencoder-like nonnegative matrix factorization for community detection". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2018, pp. 1393–1402.

[208] Xiao Wang et al. "Community Preserving Network Embedding." In: *AAAI*. 2017, pp. 203–209.

[209] Martin Rosvall and Carl T Bergstrom. "Maps of random walks on complex networks reveal community structure". In: *Proceedings of the National Academy of Sciences* 105.4 (2008), pp. 1118–1123.

[210] Martin Rosvall and Carl T Bergstrom. "Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems". In: *PloS one* 6.4 (2011), e18209.

[211] Martin Rosvall et al. "Memory in network flows and its effects on spreading dynamics and community detection". In: *Nature communications* 5.1 (2014), pp. 1–13.

[212] Christian Persson et al. "Maps of sparse Markov chains efficiently reveal community structure in network flows with memory". In: *arXiv preprint arXiv:1606.08328* (2016).

[213] Di Jin et al. "A Markov random walk under constraint for discovering overlapping communities in complex networks". In: *Journal of Statistical Mechanics: Theory and Experiment* 2011.5 (2011), P05031.

[214] Kyle Kloster and David F Gleich. "Heat kernel based community detection". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 1386–1395.

[215] Vinko Zlatić, Andrea Gabrielli, and Guido Caldarelli. "Topologically biased random walk and community finding in networks". In: *Physical Review E* 82.6 (2010), p. 66109.

[216] Jian Liu and Tingzhan Liu. "Detecting community structure in complex networks using simulated annealing with k-means algorithms". In: *Physica A: Statistical Mechanics and its Applications* 389.11 (2010), pp. 2300–2309.

[217] Hui-Jia Li et al. "Potts model based on a Markov process computation solves the community structure problem effectively". In: *Physical Review E* 86.1 (2012), p. 16109.

[218] Renaud Lambiotte and Martin Rosvall. “Ranking and clustering of nodes in networks with smart teleportation”. In: *Physical Review E* 85.5 (2012), p. 56107.

[219] Wenjun Wang et al. “Fuzzy overlapping community detection based on local random walk and multidimensional scaling”. In: *Physica A: Statistical Mechanics and its Applications* 392.24 (2013), pp. 6578–6586.

[220] Yang Yang et al. “Closed walks for community detection”. In: *Physica A: Statistical Mechanics and its Applications* 397 (2014), pp. 129–143.

[221] Lorenzo Orecchia and Zeyuan Allen Zhu. “Flow-based algorithms for local graph clustering”. In: *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*. SIAM. 2014, pp. 1267–1286.

[222] Vsevolod Salnikov, Michael T Schaub, and Renaud Lambiotte. “Using higher-order Markov models to reveal flow-based communities in networks”. In: *Scientific reports* 6 (2016), p. 23194.

[223] Dongxiao He et al. “A network-specific Markov random field approach to community detection”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[224] Rania Ibrahim and David Gleich. “Nonlinear Diffusion for Community Detection and Semi-Supervised Learning”. In: *The World Wide Web Conference*. 2019, pp. 739–750.

[225] Symeon Papadopoulos et al. “Community detection in social media”. In: *Data Mining and Knowledge Discovery* 24.3 (2012), pp. 515–554.

[226] Haoran Xie et al. “Community-aware user profile enrichment in folksonomy”. In: *Neural Networks* 58 (2014), pp. 111–121.

[227] Srijan Kumar et al. “An army of me: Sockpuppets in online discussion communities”. In: *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee. 2017, pp. 857–866.

[228] Cristian Danescu-Niculescu-Mizil et al. “No country for old members: User lifecycle and linguistic change in online communities”. In: *Proceedings of the 22nd international conference on World Wide Web*. ACM. 2013, pp. 307–318.

[229] Sanjay Ram Kairam, Dan J Wang, and Jure Leskovec. “The life and death of online groups: Predicting group growth and longevity”. In: *Proceedings of the fifth ACM international conference on Web search and data mining*. 2012, pp. 673–682.

[230] Jure Leskovec and Julian J Mcauley. “Learning to discover social circles in ego networks”. In: *Advances in neural information processing systems*. 2012, pp. 539–547.

[231] Ullas Gargi et al. “Large-scale community detection on youtube for topic discovery and exploration”. In: *Fifth International AAAI Conference on Weblogs and Social Media*. 2011.

[232] Zhongying Zhao et al. “Topic oriented community detection through social objects and link analysis in social networks”. In: *Knowledge-Based Systems* 26 (2012), pp. 164–173.

[233] Mrinmaya Sachan et al. “Using content and interactions for discovering communities in social networks”. In: *Proceedings of the 21st international conference on World Wide Web*. 2012, pp. 331–340.

[234] Nagarajan Natarajan, Prithviraj Sen, and Vineet Chaoji. “Community detection in content-sharing social networks”. In: *Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining*. 2013, pp. 82–89.

[235] Mert Ozer, Nyunsu Kim, and Hasan Davulcu. “Community detection in political Twitter networks using Nonnegative Matrix Factorization methods”. In: *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE. 2016, pp. 81–88.

[236] Yu Wang et al. “Community-based greedy algorithm for mining top-k influential nodes in mobile social networks”. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010, pp. 1039–1048.

[237] Federico Botta and Charo I del Genio. “Analysis of the communities of an urban mobile phone network”. In: *PloS one* 12.3 (2017), e0174198.

[238] Amanda L Traud et al. “Comparing community structure to characteristics in online collegiate social networks”. In: *SIAM review* 53.3 (2011), pp. 526–543.

[239] Javier O Garcia et al. “Applications of community detection techniques to brain graphs: Algorithmic considerations and implications for neural function”. In: *Proceedings of the IEEE* 106.5 (2018), pp. 846–867.

[240] Ying Liu, Jason Moser, and Selin Aviyente. “Network community structure detection for directional neural networks inferred from multichannel multisubject EEG data”. In: *IEEE Transactions on Biomedical Engineering* 61.7 (2014), pp. 1919–1930.

[241] Tiantian He and Keith CC Chan. “Evolutionary graph clustering for protein complex identification”. In: *IEEE/ACM transactions on computational biology and bioinformatics* 15.3 (2016), pp. 892–904.

[242] Tamás Nepusz, Haiyuan Yu, and Alberto Paccanaro. “Detecting overlapping protein complexes in protein-protein interaction networks”. In: *Nature methods* 9.5 (2012), p. 471.

[243] Anna CF Lewis et al. “The function of communities in protein interaction networks at multiple scales”. In: *BMC systems biology* 4.1 (2010), p. 100.

[244] Manish Gupta et al. “Evolutionary clustering and analysis of bibliographic networks”. In: *Advances in Social Networks Analysis and Mining (ASONAM), 2011 International Conference on*. IEEE. 2011, pp. 63–70.

[245] Tanmoy Chakraborty and Abhijnan Chakraborty. “OverCite: Finding overlapping communities in citation network”. In: *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*. IEEE. 2013, pp. 1124–1131.

[246] Ruiqi Hu et al. “Co-clustering enterprise social networks”. In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2016, pp. 107–114.

[247] Danielle S Bassett et al. “Extraction of force-chain network architecture in granular materials using community detection”. In: *Soft Matter* 11.14 (2015), pp. 2731–2744.

[248] Andrés E Coca and Liang Zhao. “Musical rhythmic pattern extraction using relevance of communities in networks”. In: *Information Sciences* 329 (2016), pp. 819–848.

[249] Hanyin Fang et al. “Community-based question answering via heterogeneous social network learning”. In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[250] Jure Leskovec and Andrej Krevl. “{SNAP Datasets}:{Stanford} Large Network Dataset Collection”. In: (June 2015).

[251] Jérôme Kunegis. “Konect: the koblenz network collection”. In: *Proceedings of the 22nd International Conference on World Wide Web*. 2013, pp. 1343–1350.

[252] Ryan Rossi and Nesreen Ahmed. “The network data repository with interactive graph analytics and visualization”. In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015.

[253] Jure Leskovec, Kevin J Lang, and Michael Mahoney. “Empirical comparison of algorithms for network community detection”. In: *Proceedings of the 19th international conference on World wide web*. ACM. 2010, pp. 631–640.

[254] Jaewon Yang and Jure Leskovec. “Defining and evaluating network communities based on ground-truth”. In: *Knowledge and Information Systems* 42.1 (2015), pp. 181–213.

[255] Günce Keziban Orman, Vincent Labatut, and Hocine Cherifi. “Comparative evaluation of community detection algorithms: a topological approach”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2012.8 (2012), P08001.

[256] Darko Hric, Richard K Darst, and Santo Fortunato. “Community detection in networks: Structural communities versus ground truth”. In: *Physical Review E* 90.6 (2014), p. 62805.

[257] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. “Benchmark graphs for testing community detection algorithms”. In: *Physical review E* 78.4 (2008), p. 046110.

[258] Zhao Yang, René Algesheimer, and Claudio J Tessone. “A comparative analysis of community detection algorithms on artificial networks”. In: *Scientific reports* 6 (2016), p. 30750.

[259] Bruno Abrahao et al. “On the separability of structural classes of communities”. In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2012, pp. 624–632.

[260] Meng Wang et al. “Community detection in social networks: an in-depth benchmarking study with a procedure-oriented framework”. In: *Proceedings of the VLDB Endowment* 8.10 (2015), pp. 998–1009.

[261] Hua-Wei Shen and Xue-Qi Cheng. “Spectral methods for the detection of network community structure: a comparative analysis”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2010.10 (2010), P10020.

[262] Rodrigo Aldecoa and Ignacio Marín. “Surprise maximization reveals the community structure of complex networks”. In: *Scientific reports* 3.1 (2013), pp. 1–9.

[263] Rodrigo Aldecoa and Ignacio Marín. “Deciphering network community structure by surprise”. In: *PloS one* 6.9 (2011).

[264] Vincent A Traag, Gautier Krings, and Paul Van Dooren. “Significant scales in community structure”. In: *Scientific reports* 3.1 (2013), pp. 1–10.

[265] Tanmoy Chakraborty et al. “On the permanence of vertices in network communities”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014, pp. 1396–1405.

[266] Tanmoy Chakraborty et al. “Permanence and community structure in complex networks”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 11.2 (2016), pp. 1–34.

[267] Atsushi Miyauchi and Yasushi Kawase. “What is a network community? A novel quality function and detection algorithms”. In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 2015, pp. 1471–1480.

[268] Pan Zhang. "Evaluating accuracy of community detection using the relative normalized mutual information". In: *Journal of Statistical Mechanics: Theory and Experiment* 2015.11 (2015), P11006.

[269] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. "Finding community structure in very large networks". In: *Physical review E* 70.6 (2004), p. 66111.

[270] Pascal Pons and Matthieu Latapy. "Computing communities in large networks using random walks". In: *International Symposium on Computer and Information Sciences*. Springer. 2005, pp. 284–293.

[271] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. "Deepwalk: Online learning of social representations". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014, pp. 701–710.

[272] Christopher JC Burges. "From ranknet to lambdarank to lambdamart: An overview". In: *Learning* 11.23-581 (2010), p. 81.

[273] Benjamin Paul Chamberlain et al. "Real-time community detection in full social networks on a laptop". In: *PloS one* 13.1 (2018), e0188702.

[274] Eduar Castrillo, Elizabeth León, and Jonatan Gómez. "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. 2017, pp. 982–989.

[275] Ilya Sutskever, James Martens, and Geoffrey E Hinton. "Generating text with recurrent neural networks". In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 2011, pp. 1017–1024.

[276] Piji Li et al. "Neural rating regression with abstractive tips generation for recommendation". In: *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM. 2017, pp. 345–354.

[277] Zhongqing Wang and Yue Zhang. "Opinion recommendation using a neural model". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017, pp. 1626–1637.

[278] Lawrence E Blume and David Easley. "Rationality". In: *The new Palgrave dictionary of economics* 6 (2008), pp. 884–893.

[279] Roy Brouwer, Ivana Logar, and Oleg Sheremet. "Choice consistency and preference stability in test-retests of discrete choice experiment and open-ended willingness to pay elicitation formats". In: *Environmental and Resource Economics* 68.3 (2017), pp. 729–751.

[280] Nasser M Nasrabadi. "Pattern recognition and machine learning". In: *Journal of electronic imaging* 16.4 (2007), p. 049901.

[281] Andriy Mnih and Ruslan R Salakhutdinov. "Probabilistic matrix factorization". In: *Advances in neural information processing systems*. 2008, pp. 1257–1264.

[282] Weike Pan and Li Chen. "GBPR: Group Preference Based Bayesian Personalized Ranking for One-Class Collaborative Filtering." In: *IJCAI*. Vol. 13. 2013, pp. 2691–2697.

[283] Xiangnan He et al. "Fast matrix factorization for online recommendation with implicit feedback". In: *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM. 2016, pp. 549–558.

[284] Andrea Lancichinetti and Santo Fortunato. "Limits of modularity maximization in community detection". In: *Physical review E* 84.6 (2011), p. 66122.

[285] Ju Xiang et al. "Multi-resolution modularity methods and their limitations in community detection". In: *The European Physical Journal B* 85.10 (2012), p. 352.

[286] Ju Xiang and Ke Hu. "Limitation of multi-resolution methods in community detection". In: *Physica A: Statistical Mechanics and its Applications* 391.20 (2012), pp. 4995–5003.

[287] Tatsuro Kawamoto and Yoshiyuki Kabashima. "Limitations in the spectral method for graph partitioning: Detectability threshold and localization of eigenvectors". In: *Physical Review E* 91.6 (2015), p. 62803.

[288] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. "Pointer networks". In: *Advances in neural information processing systems*. 2015, pp. 2692–2700.

[289] Jiatao Gu et al. "Incorporating copying mechanism in sequence-to-sequence learning". In: *arXiv preprint arXiv:1603.06393* (2016).

CURRICULUM VITAE

Education

08/2015–06/2020	**Ph.D. in Information Science** Indiana University Bloomington, USA
08/2013–05/2015	**Master in Information Science** University of Pittsburgh, USA
08/2009–05/2013	**Bachelor in Information Management and System** Shanghai International Studies University, China

Industry Experience

06/2019–09/2019	Data Scientist Intern, Amazon Alexa AI, USA
02/2018–03/2019	NLP research Intern, Alibaba DAMO Academy, China

Teaching Experience

01/2018–12/2019	Instructor / Lecturer, Indiana University Bloomington, USA
08/2015–12/2017	Associate Instructor, Indiana University Bloomington, USA

Publication

Zheng Gao, Hongsong Li, Zhuoren Jiang, Xiaozhong Liu. Detecting User Community in Sparse Domain via Cross-Graph Pairwise Learning. *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2020.

Zheng Gao, Lujun Zhao, Heng Huang, Hongsong Li, Changlong Sun, Luo Si, Xiaozhong Liu. Behavior based Dynamic Summarization on Product Aspects via Reinforcement Neighbour Selection. *European Conference on Artificial Intelligence (ECAI)*, 2020.

Zhuoren Jiang, **Zheng Gao**, Jinjiong Lan, Hongxia Yang, Yao Lu and Xiaozhong Liu. Task-Oriented Genetic Activation for Large-Scale Complex Heterogeneous Graph Embedding. *The Web Conference (WWW)*, 2020.

Zheng Gao, Chun Guo, Xiaozhong Liu. Efficient Personalized Community Detection via Genetic Evolution. *The Genetic and Evolutionary Computation Conference (GECCO)*, 2019.

Zheng Gao, Gang Fu, Chunping Ouyang, Satoshi Tsutsui, Xiaozhong Liu, Jeremy Yang, Christopher Gessner, Brian Foote, David Wild, Ying Ding, Qi

Yu. edge2vec: Representation Learning Using Edge Semantics for Biomedical Knowledge Discovery. *BMC Bioinformatics*, 2019. (impact factor = 2.511).

Yongzhen Wang, Xiaozhong Liu, **Zheng Gao**. Neural Related Work Summarization with a Joint Context-driven Attention Mechanism. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Zizhe Gao, **Zheng Gao**, Heng Huang, Zhuoren Jiang, Yuliang Yan. An End-to-end Model of Predicting Diverse Ranking On Heterogeneous Feeds. *eCOM Workshop at ACM SIGIR Conference on Research and Development in Information Retrieval (eCom-SIGIR)*, 2018.

Zheng Gao, Lin Guo, Chi Ma, Xiao Ma, Kai Sun, Hang Xiang, Xiaoqiang Zhu, Hongsong Li, Xiaozhong Liu. AMAD: Adversarial Multiscale Anomaly Detection on High-Dimensional and Time-Evolving Categorical Data. *Deep Learning Practice for High-Dimensional Sparse Data Workshop at ACM SIGKDD Conference on Knowledge Discovery and Data Mining (DLP-KDD)*, 2019.

Zhuoren Jiang, Liangcai Gao, Ke Yuan, **Zheng Gao**, Zhi Tang, Xiaozhong Liu. Mathematics Content Understanding for Cyberlearning via Formula Evolution Map. *ACM International Conference on Information and Knowledge Management (CIKM)*, 2018.

Xiaozhong Liu, Xing Yu, **Zheng Gao**, Tian Xia, Johan Bollen. Comparing Community-based Information Adoption and Diffusion across Different Microblogging Sites. *ACM Conference on Hypertext and Social Media*, 2016.

Zheng Gao, Vincent Malic, Shutian Ma, Patrick Shih. How to Make a Successful Movie: Factor Analysis from both Financial and Critical Perspectives. *International Conference on Information*, 2019.

Yongzhen Wang, Yan Lin, **Zheng Gao**, Yan Chen. A Two-stage Iterative Approach to Improve Crowdsourcing-based Relevance Assessment. *Arabian Journal for Science and Engineering*, 2019.

Zheng Gao, John Wolohan, Fast NLP-based Pattern Matching in Real Time Tweet Recommendation. *Text REtrieval Conference (TREC)*, 2017.

Zheng Gao, Rui Bi. University of Pittsburgh at TREC 2014 Microblog Track. *Text REtrieval Conference (TREC)*, 2014.

Zheng Gao, Xiaozhong Liu. Personalized Community Detection in Scholarly Network. *International Conference on Information*, 2017.

Tian Xia, Xing Yu, **Zheng Gao**, Yijun Gu, Xiaozhong Liu. Internal/External Information Access and Information Diffusion in Social Media. *International Conference on Information*, 2017.

Nan Li, Naren Suri, **Zheng Gao**, Tian Xia, Xiaozhong Liu, Katy Borner. Enter a Job, Get Course Recommendations. *International Conference on Informa-*

tion, 2017.

Chenwei Zhang, **Zheng Gao**, Xiaozhong Liu. How Others Affect Your Twitter #hashtag Adoption? Examination of Community-based and Context-based Information Diffusion in Twitter. *International Conference on Information*, 2015.

Zheng Gao, Patrick C. Shih. Communities of Support: Social Support Discussion in a HIV Online Forum. *International Symposium of Chinese CHI*, 2019.

Satoshi Tsutsui, **Zheng Gao**, Yuzhuo Wang, Guilin Meng, Ying Ding. A Case Study on Viziometrics: What's the Role of Western Blots in Alzheimer's Disease Literature. *International Conference on Information*, 2018.