Broad Learning: An Emerging Area in Social Network Analysis

Jiawei Zhang
IFM Lab
Florida State University
Tallahassee, FL 32311, USA
jiawei@ifmlab.org

Philip S. Yu
BDSC Lab
University of Illinois at Chicago
Chicago, IL 60607, USA
psyu@cs.uic.edu

ABSTRACT

Looking from a global perspective, the landscape of online social networks is highly fragmented. A large number of online social networks have appeared, which can provide users with various types of services. Generally, information available in these online social networks is of diverse categories, which can be represented as heterogeneous social networks (HSNs) formally. Meanwhile, in such an age of online social media, users usually participate in multiple online social networks simultaneously, who can act as the anchors aligning different social networks together. So multiple HSNs not only represent information in each social network, but also fuse information from multiple networks.

Formally, the online social networks sharing common users are named as the aligned social networks, and these shared users are called the anchor users. The heterogeneous information generated by users' social activities in the multiple aligned social networks provides social network practitioners and researchers with the opportunities to study individual user's social behaviors across multiple social platforms simultaneously. This paper presents a comprehensive survey about the latest research works on multiple aligned HSNs studies based on the broad learning setting, which covers 5 major research tasks, including *network alignment*, *link prediction*, *community detection*, *information diffusion* and *network embedding* respectively.

Keywords

Broad Learning; Heterogeneous Social Networks; Network Alignment; Link Prediction; Community Detection; Information Diffusion; Network Embedding

1. INTRODUCTION

In the real world, on the same information entities, e.g., products, movies, POIs (points-of-interest) and even human beings, a large amount of information can actually be collected from various sources. These sources are usually of different varieties, like Walmart vs Amazon for commercial products; IMDB vs Rotten Tomatoes for movies; Yelp vs Foursquare for POIs; and various online social networks websites for social media users. Each information source provides a specific signature of the same entity from a unique underlying aspect. However, in many cases, these information sources are usually separated from each other, and an effective fusion of these different information sources provides an opportunity for researchers and practitioners to understand the entities more comprehensively, which renders *broad learning* [93; 85; 108] an extremely important learning task.

Broad learning introduced in [93; 85; 108] is a new type of learning task, which focuses on fusing multiple large-scale information sources of diverse varieties together and carrying out synergistic data mining tasks across these fused sources in one unified analytic. Fusing and mining multiple information sources of large volumes and diverse varieties are the fundamental problems in big data studies. Broad learning investigates the principles, methodologies and algorithms for synergistic knowledge discovery across multiple aligned information sources, and evaluates the corresponding benefits. Great challenges exist in broad learning for the effective fusion of relevant knowledge across different aligned information sources, which depends upon not only the relatedness of these information sources, but also the target application problems to be studied. Broad learning aims at developing general methodologies, which will be shown to work for a diverse set of applications, while the specific parameter settings can be learned for each application from the training data.

In this paper, we will focus on introducing the broad learning research works done based on online social media data. Nowadays, to enjoy more social network services, people are usually involved in multiple online social networks simultaneously, such as Facebook, Twitter and Foursquare [104; 32]. Individuals usually have multiple separate accounts in different social networks, and discovering the correspondence between accounts of the same user (i.e., network alignment or user anchoring) [98; 99; 32; 91; 96; 84] will be an interesting problem. What's more, network alignment is also the crucial prerequisite step for many interesting inter-network synergistic knowledge discovery applications, like (1) inter-network link prediction/recommendation [94; 104; 86; 87; 96; 84; 25; 100; 88], (2) mutual community detection [95; 26; 97; 57; 85; 101], (3) cross-platform information diffusion [78; 77; 103], and (4) multiple networks synergistic embedding [83; 93]. These application tasks are fundamental problems in social network studies, which together with the network alignment problem will form the backbone of the multiple social network broad learning ecosystem.

This paper will cover five strongly correlated social network research problems based on broad learning:

- **Network Alignment**: Identifying the common users shared by different online social networks can effectively combine these networks together, which will also provide the opportunity to study users' social behaviors from a more comprehensive perspective. Many research works have proposed to align the online social networks together by inferring the mappings of the shared users between different networks, which will be introduced in a great detail in this paper.
- **Link Prediction**: Users' friendship connections in different networks have strong correlations. With the social activity data across multiple aligned social networks, we can acquire more knowledge about users and their personal social prefer-

ences. We will introduce the existing research works on the social link prediction problem across multiple aligned social sites simultaneously.

- **Community Detection**: Information available across multiple aligned social networks provides more complete signals revealing the social communities formed by people in the real world. We will introduce the existing works on community detection with knowledge fused from multiple aligned heterogeneous social networks as the third task.
- **Information Diffusion**: The formulation of multiple aligned heterogeneous social networks provides researchers with the opportunity to study the information diffusion process across different social sites. The latest research papers on cross-network information diffusion will be illustrated as well.
- **Network Embedding**: Information from other aligned networks can help refine the feature representations of users effectively. In recent years, some research papers introduce the synergistic network embedding across aligned social networks, where knowledge from other external networks can be effectively utilized in representation learning tasks.

The remainder parts of this paper will be organized as follows. We will first provide the basic terminology definitions in Section 2. Via the anchor links, we will introduce the inter-network meta path concept in Section 3, which will be used in the following sections. The network alignment research papers will be introduced in Section 4. Inter-network link prediction and friend recommendation will be talked about in Section 5. A detailed review about cross-network community detection will be provided in Section 6. Broad learning based information diffusion is introduced in Section 7 and network embedding works are available in Section 8. Finally, we will illustrate several potential future development directions about broad learning and conclude this paper in Section 9.

2. TERMINOLOGY DEFINITION

Online social networks (OSNs) denote the online platforms which allow people to build social connections with other people, who share similar personal or career interests, backgrounds, and real-life connections. Online social networking sites vary greatly and each category of online social networks can provide a specific type of featured services. To enjoy different kinds of social networks services simultaneously, users nowadays are usually involved in many of these online social networks aforementioned at the same time, in each of which they will form separate social connections and generate a large amount of social information.

Formally, the online social networks can be represented as graphs. Besides the users, there usually exist many other types of information entities, like posts, photos, videos and comments, generated by users' online social activities. Information entities in online social networks are extensively connected, and the connections among different types of nodes usually have different physical meanings. The diverse nodes and connections render the online social networks a very complex graph structure. Meanwhile, depending on the categories of information entities and connections involved, the online social networks can be divided into different types, like homogeneous network, bipartite network and heterogeneous network. To model the phenomenon that users are involved multiple networks, a new concept called "multiple aligned heterogeneous social networks" [104; 32] has been proposed in recent years.

For the networks with simple structures, like the homogeneous networks merely involving users and friendship links, the social patterns in them are usually easy to study. However, for the networks with complex structures, like the heterogeneous networks, the nodes can be connected by different types of link, which will have totally different physical meanings. One general technique for heterogeneous network studies is "meta path" [65; 104], which specifically depicts certain link-sequence structures connecting node defined based on the network schema. The meta path concept can also be extended to the multiple aligned social network scenario as well, which can connect the node across different social networks. Given a network $G = (\mathcal{V}, \mathcal{E})$, we can represent the set of node and link types involved in the network as sets $\mathcal{N}$ and $\mathcal{R}$ respectively. Based on such information, the *social network* concept can be formally defined based on the graph concept by adding the mappings indicating the node and link type information.

DEFINITION 1. *(Social Networks): Formally, a heterogeneous social network can be represented as $G = (\mathcal{V}, \mathcal{E}, \phi, \psi)$, where $\mathcal{V}$, $\mathcal{E}$ are the sets of nodes and links in the network, and mappings $\phi : \mathcal{V} \to \mathcal{N}$, $\psi : \mathcal{E} \to \mathcal{R}$ project the nodes and links to their specific types respectively. In many cases, the mappings ϕ, ψ are omitted assuming that the node and link types are known by default.*

In the following parts of this paper, depending on the categories of information involved in the online social networks, we propose to categorize the online social networks into three groups: *homogeneous social networks*, *heterogeneous social networks* and *aligned heterogeneous social networks*.

2.1 Homogeneous Social Network

DEFINITION 2. *(Homogeneous Social Network): For a online social network G, if there exists one single type of nodes and links in the network (i.e., $|\mathcal{N}| = |\mathcal{R}| = 1$), then the network is called a homogeneous social network.*

Given a *homogeneous social network* $G = (\mathcal{V}, \mathcal{E})$ with user set $\mathcal{V}$ and social relationship set $\mathcal{E}$, depending on whether the links in G are directed or undirected, the social link can denote either the *follow* links or *friendship* links among individuals. Given an individual user $u \in \mathcal{V}$ in an undirected friendship social network, the set of users connected to u can be represented as the friends of user u in the network G, denoted as $\Gamma(u) \subset \mathcal{V} = \{v | v \in \mathcal{V} \wedge (u, v) \in \mathcal{E}\}$. The number of friends that user u has in the network is also called the degree of node u, i.e., $|\Gamma(u)|$.

Meanwhile, in a directed network G, the set individuals followed by u (i.e., $\Gamma_{out}(u) = \{v | v \in \mathcal{V} \wedge (u, v) \in \mathcal{E}\}$) are called the set of followees of u; and the set of individuals that follow u (i.e., $\Gamma_{out}(u) = \{v | v \in \mathcal{V} \wedge (v, u) \in \mathcal{E}\}$) are called the set of followers of u. The number of users who follow u is called the in-degree of u, and the number of users followed by u is called the out-degree of u in the network. For the users with large out-degrees, they are called the *hubs* [31] in the network; while those with large in-degrees, they are called the *authorities* [31] in the network.

2.2 Heterogeneous Social Network

DEFINITION 3. *(Heterogeneous Social Network): For an online social network G, if there exists multiple types of nodes or links in the network (i.e., $|\mathcal{N}| > 1$, or $|\mathcal{R}| > 1$), then the network is called a heterogeneous social network.*

Most of the online networks in the real world may contain very complex information involving multiple types of nodes and connections. For instance, in the social networks to be studied in the following part, they may involve users, posts, check-ins, words and timestamps, as well as the friendship links, write links and contain links among these nodes. Formally, such an online social network can be defined as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of nodes and $\mathcal{E}$ represents the set of links in G. The node set $\mathcal{V}$ can be divided into several subsets $\mathcal{V} = \mathcal{U} \cup \mathcal{P} \cup \mathcal{L} \cup \mathcal{T} \cup \mathcal{W}$ involving the user nodes, post nodes, location nodes, word nodes and timestamp nodes respectively. The link set $\mathcal{E}$ can be divided into several subsets as well, $\mathcal{E} = \mathcal{E}_{u,u} \cup \mathcal{E}_{u,p} \cup \mathcal{E}_{p,l} \cup \mathcal{E}_{p,w} \cup \mathcal{E}_{p,t}$, containing the links among users, the links between users and posts, and those connecting posts with location checkins, words, and timestamps.

In the *heterogeneous social networks*, each node can be connected with a set of nodes belonging to different categories via various type of connections. For example, given a user $u \in \mathcal{U}$, the set of user node incident to u via the friend links can be represented as the online friends of u, denoted as set $\{v | v \in \mathcal{U}, (u, v) \in \mathcal{E}_{u,u}\}$; the set of post node incident to u via the write links can be represented as the posts written by u, denoted as set $\{w | w \in \mathcal{P}, (u, w) \in \mathcal{E}_{u,p}\}$. The location check-in nodes, word nodes and timestamp nodes are not directly connected to the user node, while via the post nodes, we can also obtain the set of locations/words/timestamps that are visited/used/active-at by user u in the network. Such indirect connections can be described more clearly by the *meta path* concept more clearly in Section 3.

2.3 Aligned Heterogeneous Social Networks

Definition 4. *(Multiple Aligned Heterogeneous Networks): Formally, the multiple aligned heterogeneous networks involving n networks can be defined as $\mathcal{G} = ((G^{(1)}, G^{(2)}, \cdots, G^{(n)}),$ $(\mathcal{A}^{(1,2)}, \mathcal{A}^{(1,3)}, \cdots, \mathcal{A}^{(n-1,n)}))$, where $G^{(1)}, G^{(2)}, \cdots, G^{(n)}$ denote these n heterogeneous social networks and the sets $\mathcal{A}^{(1,2)}$, $\mathcal{A}^{(1,3)}, \cdots, \mathcal{A}^{(n-1,n)}$ represent the undirected anchor links aligning these networks respectively.*

Anchor links actually refer to the mappings of information entities shared across different sources, which correspond to the the same information entity in the real world, e.g., users in online social networks, authors in different bibliographic networks, and movies in the movie knowledge libraries.

Definition 5. *(Anchor Link): Given two heterogeneous networks $G^{(i)}$ and $G^{(j)}$ which share some common information entities, the set of anchor links connecting $G^{(i)}$ and $G^{(j)}$ can be represented as set $\mathcal{A}^{(i,j)} = \{(u^{(i)}_m, u^{(j)}_n) | u^{(i)}_m \in \mathcal{V}^{(i)} \wedge u^{(j)}_n \in \mathcal{V}^{(j)} \wedge u^{(i)}_m, u^{(j)}_n$ denote the same information entity$\}$.*

The *anchor links* depict a transitive relationship among the information entities across different networks. Given 3 information entities $u^{(i)}_m$, $u^{(j)}_n$, $u^{(k)}_o$ from networks $G^{(i)}$, $G^{(j)}$ and $G^{(k)}$ respectively, if $u^{(i)}_m$, $u^{(j)}_n$ are connected by an anchor link and $u^{(j)}_n$, $u^{(k)}_o$ are connected by another anchor link, then the user pair $u^{(i)}_m$, $u^{(k)}_o$ will be connected by an anchor link by default. For more detailed definitions about other related terms, like *anchor users*, *non-anchor users*, *full alignment*, *partial alignment* and *non-alignment*, please refer to [104].

3. META PATH

To deal with the social networks, especially the heterogeneous social networks, a very useful technique is *meta paths* [65; 104]. *Meta path* is a concept defined based on the network schema, outlining the connections among nodes belonging to different categories. For the nodes which are not directly connected, their relationships can be depicted with the meta path concept. In this part, we will define the meta path concept, and introduce a set of meta paths within and across real-world heterogeneous social networks respectively.

3.1 Network Schema

Given a network $G = (\mathcal{V}, \mathcal{E})$, we can define its *network schema* to describe the categories of nodes and links involved in G.

Definition 6. *(Network Schama): Formally, the schema of network G can be denoted as $S_G = (\mathcal{N}, \mathcal{R})$, where $\mathcal{N}$ and $\mathcal{R}$ are the sets of node type and link type in the network respectively.*

Network schema provides a meta level description of networks. Meanwhile, if a network G can be outlined by the network schema S_G, G is also called a *network instance* of the network schema. For a given node $u \in \mathcal{V}$, we can represent its corresponding node type as $\phi(u) = N \in \mathcal{N}$, and call u is an instance of node type N, which can also be denoted as $u \in N$ for simplicity. Similarly, for a link (u, v), we can denotes its link type as $\psi((u, v)) = R \in \mathcal{R}$, or $(u, v) \in R$ for short. The inverse relation R^{-1} denotes a new link type with reversed direction. Generally, R is not equal to R^{-1}, unless R is symmetric.

3.2 Meta Path in Heterogeneous Social Networks

Meta path is a concept defined based on the network schema denoting the correlation of nodes based on the heterogeneous information (i.e., different types of nodes and links) in the networks.

Definition 7. *(Meta Path): A meta path P defined based on the network schema $S_G = (\mathcal{N}, \mathcal{R})$ can be represented as $P = N_1 \xrightarrow{R_1} N_2 \xrightarrow{R_2} \cdots N_{k-1} \xrightarrow{R_{k-1}} N_k$, where $N_i \in \mathcal{N}, i \in \{1, 2, \cdots, k\}$ and $R_i \in \mathcal{R}, i \in \{1, 2, \cdots, k-1\}$.*

Furthermore, depending on the categories of node and link types involved in the meta path, we can specify the meta path concept into several more refined groups, like *homogeneous meta path* and *heterogeneous meta path*, or *social meta path* and other *meta paths*.

Definition 8. *(Homogeneous/Heterogeneous Meta Path): Let $P = N_1 \xrightarrow{R_1} N_2 \xrightarrow{R_2} \cdots N_{k-1} \xrightarrow{R_{k-1}} N_k$ denote a meta path defined based on the network schema $S_G = (\mathcal{N}, \mathcal{R})$. If all the node types and link types involved in P are of the same category, P is called a homogeneous meta path; otherwise, P is called a heterogeneous meta path.*

The meta paths can connect any kinds of node type pairs, and specifically, for the meta paths starting and ending with the user node types, those meta paths are called the *social meta paths*.

Definition 9. *(Social Meta Path): Let $P = N_1 \xrightarrow{R_1} N_2 \xrightarrow{R_2} \cdots N_{k-1} \xrightarrow{R_{k-1}} N_k$ denote a meta path defined based on network schema $S_G = (\mathcal{N}, \mathcal{R})$. If the starting and ending node types N_1 and N_k are both the user node type, P is called a social meta path.*

Users are usually the focus in social network studies, and the *social meta paths* are frequently used in both research and real-world applications and services. The number of path segments in the meta path is called the meta path length. For instance, the length of meta path $P = N_1 \xrightarrow{R_1} N_2 \xrightarrow{R_2} \cdots N_{k-1} \xrightarrow{R_{k-1}} N_k$ is $k-1$. Meta paths can also been concatenated together with the *meta path composition operator*.

Definition 10. *(Meta Path Composition): Meta paths $P^1 = N^1_1 \xrightarrow{R^1_1} N^1_2 \xrightarrow{R^1_2} \cdots N^1_{k-1} \xrightarrow{R^1_{k-1}} N^1_k$, and $P^2 = N^2_1 \xrightarrow{R^2_1} N^2_2 \xrightarrow{R^2_2} \cdots N^2_{l-1} \xrightarrow{R^2_{l-1}} N^1_l$ can be concatenated together to form a longer meta path $P = P^1 \circ P^2 = N^1_1 \xrightarrow{R^1_1} \cdots \xrightarrow{R^1_{k-1}} N^1_k \xrightarrow{R^2_1} N^2_2 \xrightarrow{R^2_2} \cdots N^2_{l-1} \xrightarrow{R^2_{l-1}} N^1_l$, if the ending node type of P^1 is the same as the starting node type of P^2, i.e., $N^1_k = N^2_1$. The new composed meta path is of length $k + l - 2$.*

Meta path $P = N_1 \xrightarrow{R_1} N_2 \xrightarrow{R_2} \cdots N_{k-1} \xrightarrow{R_{k-1}} N_k$ can also been treated as the concatenation of simple meta paths $N_1 \xrightarrow{R_1} N_2$, $N_2 \xrightarrow{R_2} N_3, \cdots, N_{k-1} \xrightarrow{R_{k-1}} N_k$, which can be represented as $P = R_1 \circ R_2 \circ \cdots \circ R_{k-1} \circ R_k$.

3.3 Meta Path across Aligned Heterogeneous Social Networks

Besides the meta paths within one single heterogeneous network, the meta paths can also be defined across multiple aligned heterogeneous networks via the *anchor meta paths*.

DEFINITION 11. *(Anchor Meta Path): Let $G^{(1)}$ and $G^{(2)}$ be two heterogeneous networks sharing the common anchor information entity of types $N^{(1)} \in \mathcal{N}^{(1)}$ and $N^{(2)} \in \mathcal{N}^{(2)}$ respectively. The anchor meta path between the schemas of networks $G^{(1)}$ and $G^{(2)}$ can be represented as $\Phi = N^{(1)} \xleftrightarrow{Anchor} N^{(2)}$ of length* 1.

The *anchor meta path* is the simplest meta path across aligned networks, and a set of inter-network meta paths can be defined based on the intra-network meta paths and the anchor meta path.

DEFINITION 12. *(Inter-Network Meta Path): A meta path $\Psi = N_1 \xrightarrow{R_1} N_2 \xrightarrow{R_2} \cdots N_{k-1} \xrightarrow{R_{k-1}} N_k$ is called an inter-network meta path between networks $G^{(1)}$ and $G^{(2)}$ iff $\exists m \in \{1, 2, \cdots, k-1\}, R_m = Anchor$.*

The *inter-network meta paths* can be viewed as a composition of *intra-network meta paths* and the *anchor meta path* via the user node types. An *inter-network meta path* can be a meta path starting with an *anchor meta path* followed by the *intra-network meta paths*, or those with *anchor meta paths* in the middle. Here, we introduce several categories *inter-network meta paths* involving the anchor meta paths at different positions as defined in [104]:

- $\Psi(G^{(1)}, G^{(2)}) = \Phi(G^{(1)}, G^{(2)})$, which denotes the simplest *inter-network meta path* composed of the anchor meta path only between networks $G^{(1)}$ and $G^{(2)}$.
- $\Psi(G^{(1)}, G^{(2)}) = \Phi(G^{(1)}, G^{(2)}) \circ P(G^{(2)})$, which denotes the *inter-network meta path* starting with an *anchor meta path* and followed by the *intra-network social meta path* in network $G^{(2)}$.
- $\Psi(G^{(1)}, G^{(2)}) = P(G^{(1)}) \circ \Phi(G^{(1)}, G^{(2)})$, which denotes the *inter-network meta path* starting with the *intra-network social meta path* in network $G^{(1)}$ followed by an *anchor meta path* between networks $G^{(1)}$ and $G^{(2)}$.
- $\Psi(G^{(1)}, G^{(2)}) = P(G^{(1)}) \circ \Phi(G^{(1)}, G^{(2)}) \circ P(G^{(2)})$, which denotes the *inter-network meta path* starting and ending with the *intra-network social meta path* in networks $G^{(1)}$ and $G^{(2)}$ respectively connected by an *anchor meta path* between networks $G^{(1)}$ and $G^{(2)}$.

These meta path concepts introduced in this section will be widely used in various social network learning tasks to be introduced later.

4. NETWORK ALIGNMENT

Network alignment is an important research problem and dozens of papers have been published on this topic in the past decades. Depending on specific disciplines, the studied networks can be social networks in data mining [98; 99; 32; 91; 96; 84] protein-protein interaction (PPI) networks and gene regulatory networks in bioinformatics [27; 60; 38; 61], chemical compound in chemistry [63], data schemas in data warehouse [45], ontology in web semantics [14], graph matching in combinatorial mathematics [44], as well as graphs in computer vision [11; 6].

In bioinformatics, the network alignment problem aims at predicting the best mapping between two biological networks based on the similarity of the molecules and their interaction patterns. By studying the cross-species variations of biological networks, network alignment problem can be applied to predict conserved functional modules [58] and infer the functions of proteins [50]. Graemlin [17] conducts pairwise network alignment by maximizing an objective function based on a set of learned parameters. Some works have been done on aligning multiple network in bioinformatics. IsoRank proposed in [62] can align multiple networks greedily based on the pairwise node similarity scores calculated with spectral graph theory. IsoRankN [38] further extends IsoRank model by exploiting a spectral clustering scheme in the framework.

In recent years, with rapid development of online social networks, researchers' attention starts to shift to the alignment of social networks. Enlightened by the homogeneous network alignment method in [70], Koutra et al. [35] propose to align two bipartite graphs with a fast alignment algorithm. Zafarani et al. [76] propose to match users across social networks based on various node attributes, e.g., username, typing patterns and language patterns etc. Kong et al. formulate the heterogeneous social network alignment problem as an anchor link prediction problem. A two-step supervised network alignment method MNA is proposed in [32] to infer potential anchor links across networks with heterogeneous information in the networks. However, social networks in the real world are mostly partially aligned actually and lots of users are not anchor users. Zhang et al. have proposed a partial network alignment method specifically in [91].

In the social network alignment model building, the anchor links are very expensive to label manually, and achieving a large-sized anchor link training set can be extremely challenging. In [96], Zhang et al. propose to study the network alignment problem based on the PU (Positive and Unlabeled) learning setting instead, where the model is built based on a small amount of positive set and a large unlabeled set. Furthermore, in the case when no training data is available, via inferring the potential anchor user mappings across networks, Zhang et al. have introduced an unsupervised network alignment models for multiple (more than 2) social networks in [98] and an unsupervised network concurrent alignment model via multiple shared information entities simultaneously in [99].

Next, we will introduce the social network alignment methods based on the *pairwise* and *global* alignment settings respectively.

4.1 Pairwise Unsupervised Network Alignment

In this part, we will study the network alignment problem based on unsupervised learning setting, which needs no labeled training data. Given two heterogeneous online social networks, which can be represented as $G^{(1)} = (\mathcal{V}^{(1)}, \mathcal{E}^{(1)})$ and $G^{(2)} = (\mathcal{V}^{(2)}, \mathcal{E}^{(2)})$ respectively, the unsupervised network alignment problem aims at inferring the anchor links between networks $G^{(1)}$ and $G^{(2)}$. Let $\mathcal{U}^{(1)} \subset \mathcal{V}^{(1)}$ and $\mathcal{U}^{(2)} \subset \mathcal{V}^{(2)}$ be the user set in these two networks respectively, we can represent the set of potential anchor links between networks $G^{(1)}$ and $G^{(2)}$ as $\mathcal{A} = \mathcal{U}^{(1)} \times \mathcal{U}^{(2)}$. In the unsupervised network alignment problem, among all the potential anchor links in set $\mathcal{A}$, we want to infer which one exists in the real world.

Given two homogeneous networks $G^{(1)}$ and $G^{(2)}$, mapping the nodes between them is an extremely challenging task, which is also called the graph isomorphism problem [54; 18]. The graph isomorphism has been shown to be NP, but it is still not known whether it also belongs to P or NP-complete yet. So far, no efficient algorithm exists that can address the problem in polynomial time. In this part, we will introduce several heuristics based methods to solve the pairwise homogeneous network alignment problem.

4.1.1 Heuristics based Network Alignment Model

The information generated by users' online social activities can indicate their personal characteristics. The features introduced in the previous subsection, like ECN, EJC and EAA based on social connection information, similarity/distance measures based on location checkin information, temporal activity closeness, and text word usage similarity can all be used as the predictors indicating whether the cross-network user pairs are the same user or not. Besides these measures, in this part, we will introduce a category new measures, Relative Centrality Difference (RCD), which can also be applied to solve the unsupervised network alignment problem.

The centrality concept can denote the importance of users in the online social networks. Here, we assume that important users in one social network (like celebrities, movie stars and politicians) will be important as well in other networks. Based on such an assumption, the centrality of users in different networks can be an important signal for inferring the anchor links across networks.

DEFINITION 13. *(Relative Centrality Difference): Given two users $u_i^{(1)}$, $u_j^{(2)}$ from networks $G^{(1)}$ and $G^{(2)}$ respectively, let $C(u_i^{(1)})$ and $C(u_j^{(2)})$ denote the centrality scores of the users, we can define the relative centrality difference (RCD) as*

$$RCD(u_i^{(1)}, u_j^{(2)}) = \left(1 + \frac{|C(u_i^{(1)}) - C(u_j^{(2)})|}{\left(C(u_i^{(1)}) + C(u_j^{(2)})\right)/2}\right)^{-1}. \quad (1)$$

Depending on the centrality measures applied, different types of *relative centrality difference* measures can be defined. For instance, if we use node degree as the centrality measure, the *relative degree difference* can be represented as

$$RDD(u_i^{(1)}, u_j^{(2)}) = \left(1 + \frac{|D(u_i^{(1)}) - D(u_j^{(2)})|}{\left(D(u_i^{(1)}) + D(u_j^{(2)})\right)/2}\right)^{-1}. \quad (2)$$

Meanwhile, if the PageRank scores of the nodes are used to define their centrality, we can represent the relative centrality difference measure as

$$RCD(u_i^{(1)}, u_j^{(2)}) = \left(1 + \frac{|S(u_i^{(1)}) - S(u_j^{(2)})|}{\left(S(u_i^{(1)}) + S(u_j^{(2)})\right)/2}\right)^{-1}. \quad (3)$$

In the above equations, $D(u)$ and $S(u)$ denote the *node degree* and *page rank score* of node u within each network respectively.

4.1.2 IsoRank

Model IsoRank [62] initially proposed to align the biomedical networks, like protein protein interaction (PPI) networks and gene expression networks, can be used to solve the unsupervised social network alignment problem as well. The IsoRank algorithm has two stages. It first associates a score with each possible anchor links between nodes of the two networks. For instance, we can denote $r(u_i^{(1)}, u_j^{(2)})$ as the reliability score of an potential anchor link $(u_i^{(1)}, u_j^{(2)})$ between the networks $G^{(1)}$ and $G^{(2)}$, and all such scores can be organized into a vector $\mathbf{r}$ of length $|\mathcal{U}^{(1)}| \times |\mathcal{U}^{(2)}|$. In the second stage of IsoRank, it constructs the mapping for the networks by extracting from $\mathbf{r}$.

DEFINITION 14. *(Reliability Score): The reliability score $r(u_i^{(1)}, u_j^{(2)})$ of anchor link $(u_i^{(1)}, u_j^{(2)})$ is highly correlated with the support provided by the mapping scores of the neighborhoods of users $u_i^{(1)}$ and $u_j^{(2)}$. Therefore, we can define $r(u_i^{(1)}, u_j^{(2)})$ as*

$$r(u_i^{(1)}, u_j^{(2)}) \quad (4)$$
$$= \sum_{u_m^{(1)} \in \Gamma(u_i^{(1)})} \sum_{u_n^{(2)} \in \Gamma(u_i^{(2)})} \frac{1}{|\Gamma(u_i^{(1)})||\Gamma(u_j^{(2)})|} r(u_m^{(1)}, u_n^{(2)}), \quad (5)$$

where sets $\Gamma(u_i^{(1)})$ and $\Gamma(u_i^{(2)})$ represent the neighborhoods of users $u_i^{(1)}$ and $u_i^{(1)}$ respectively in networks $G^{(1)}$ and $G^{(2)}$.

If the networks are weighted, and all the intra-network connections like $(u_i^{(1)}, u_m^{(1)})$ will be associated with a weight $w(u_i^{(1)}, u_m^{(1)})$, we can represented the reliability measure of $r(u_i^{(1)}, u_j^{(2)})$ in the weighted network as

$$r(u_i^{(1)}, u_j^{(2)}) = \sum_{u_m^{(1)} \in \Gamma(u_i^{(1)})} \sum_{u_n^{(2)} \in \Gamma(u_i^{(2)})} w(u_i^{(1)}, u_j^{(2)}) r(u_m^{(1)}, u_n^{(2)}), \quad (6)$$

where the weight term

$$w(u_i^{(1)}, u_j^{(2)}) \quad (7)$$
$$= \frac{w(u_i^{(1)}, u_m^{(1)}) w(u_j^{(2)}, u_n^{(2)})}{\sum_{u_p^{(1)} \in \Gamma(u_i^{(1)})} w(u_i^{(1)}, u_p^{(1)}) \sum_{u_q^{(2)} \in \Gamma(u_i^{(2)})} w(u_j^{(2)}, u_q^{(2)})}. \quad (8)$$

As we can see, Equation 4 is a special case of Equation 6 with link weight $w(u_i^{(1)}, u_j^{(1)}) = 1$ for $u_i^{(1)} \in \mathcal{U}^{(1)}$ and $u_j^{(2)} \in \mathcal{U}^{(2)}$. Equation 4 can also be rewritten with linear algebra

$$\mathbf{r} = \mathbf{A}\mathbf{r}, \quad (9)$$

where matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{U}^{(1)}||\mathcal{U}^{(2)}| \times |\mathcal{U}^{(1)}||\mathcal{U}^{(2)}|}$ with entry

$$A((i,j),(p,q)) \quad (10)$$
$$= \begin{cases} \frac{1}{|\Gamma(u_i^{(1)})||\Gamma(u_j^{(2)})|}, & \text{if } (u_i^{(1)}, u_p^{(1)}) \in \mathcal{E}^{(1)}, (u_j^{(2)}, u_q^{(2)}) \in \mathcal{E}^{(2)}, \\ 0, & \text{otherwise}. \end{cases} \quad (11)$$

The matrix $\mathbf{A}$ is of dimension $|\mathcal{U}^{(1)}||\mathcal{U}^{(2)}| \times |\mathcal{U}^{(1)}||\mathcal{U}^{(2)}|$, where the row and column indexes correspond to different potential anchor links across the networks. The entry $A\big((i,j),(p,q)\big)$ corresponds the anchor links $(u_i^{(1)}, u_j^{(2)})$ and $(u_p^{(1)}, u_q^{(2)})$. As we can see, the above equation denotes a random walk across the graphs $G^{(1)}$ and $G^{(2)}$ via the social links and anchor links in them. The solution to the above equation denotes the principal eigenvector of the matrix $\mathbf{A}$ corresponding to the eigenvalue 1. For more information about the random walk model, please refer to [62].

4.1.3 Matrix Inference based Network Alignment

Formally, given a homogeneous network $G^{(1)}$, its structure can be organized as the adjacency matrix $\mathbf{A}_{G^{(1)}} \in \mathbb{R}^{|\mathcal{U}^{(1)}| \times |\mathcal{U}^{(1)}|}$. If network $G^{(1)}$ is unweighted, then matrix $\mathbf{A}_{G^{(1)}}$ will be a binary matrix and entry $A_{G^{(1)}}(i,p) = 1$ (or $A_{G^{(1)}}(u_i^{(1)}, u_p^{(1)}) = 1$) iff the correspond social link $(u_i^{(1)}, u_p^{(1)})$ exists. In the case that the network is weighted, the entries like $A_{G^{(1)}}(i,p) = 1$ denotes the weight of link $(u_i^{(1)}, u_p^{(1)})$ and 0 if $(u_i^{(1)}, u_p^{(1)})$ doesn't exist. In a similar way, we can also represent the social adjacency matrix $\mathbf{A}_{G^{(2)}}$ for network $G^{(2)}$ as well.

The network alignment problem aims at inferring an one-to-one node mapping function, that can project nodes from one network to the other networks. For instance, we can denote the mapping between networks $G^{(1)}$ to $G^{(2)}$ as $f : \mathcal{U}^{(1)} \to \mathcal{U}^{(2)}$. Via the mapping f, besides the nodes, the network structure can be projected across networks as well. For instance, given a social connection $(u_i^{(1)}, u_p^{(1)})$ in $G^{(1)}$, we can represent its corresponding connection in $G^{(2)}$ as $(f(u_i^{(1)}), f(u_p^{(1)}))$.

Via the mapping f, we can denote the network structure differences between $G^{(1)}$ and $G^{(2)}$ as the summation of the link projection difference between them

$$L(G^{(1)}, G^{(2)}, f) = \quad (12)$$
$$\sum_{u_i^{(1)} \in \mathcal{U}^{(1)}} \sum_{u_p^{(1)} \in \mathcal{U}^{(1)}} \left(A_{G^{(1)}}(u_i^{(1)}, u_p^{(1)}) - A_{G^{(1)}}(f(u_i^{(1)}), f(u_p^{(1)}))\right)^2. \quad (13)$$

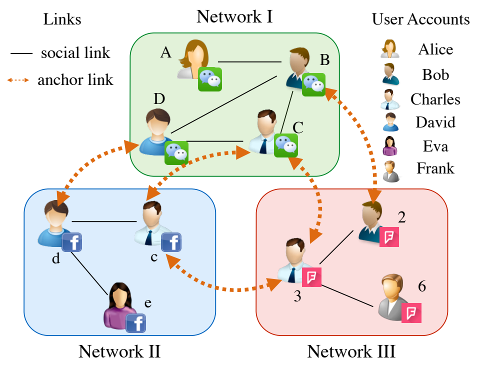

Figure 1: An example of multiple anonymized partially aligned social networks.

Formally, the one-to-one projection can be represented as a matrix $\mathbf{P}$ as well, where entry $P(i,j) = 1$ iff anchor link $(u_i^{(1)}, u_j^{(2)})$ exists between networks $G^{(1)}$ and $G^{(2)}$. Via the matrix $\mathbf{P}$, we can represent the above loss term as

$$L(\mathbf{A}_{G^{(1)}}, \mathbf{A}_{G^{(2)}}, \mathbf{P}) = \left\| \mathbf{P}^\top \mathbf{A}_{G^{(1)}} \mathbf{P} - \mathbf{A}_{G^{(2)}} \right\|^2. \quad (14)$$

If there exists a perfect mapping of users across networks, we can obtain a mapping matrix $\mathbf{P}$ introducing zero loss in the above function, i.e., $L(\mathbf{A}_{G^{(1)}}, \mathbf{A}_{G^{(2)}}, \mathbf{P}) = 0$. Inferring the optimal mapping matrix $\mathbf{P}$ which can introduce the minimum loss can be represented as the following objective function

$$\mathbf{P}^* = \arg\min_{\mathbf{P}} \left\| \mathbf{P}^\top \mathbf{A}_{G^{(1)}} \mathbf{P} - \mathbf{A}_{G^{(2)}} \right\|^2, \quad (15)$$

where the matrix $\mathbf{P}$ is usually subject to some constraint, like $\mathbf{P}$ is binary and each row and column should contain at most one entry being filled with value 1.

In general, it is not easy to find the optimal solution to the above objective function, as it is a purely combinatorial problem. Identifying the optimal solution requires the enumeration of all the potential user mapping across different networks. In [70], Umeyama provides an algorithm that can solve the function with a nearly optimal solution.

4.2 Global Unsupervised Network Alignment

The works introduced in the previous section are all about pairwise network alignment, which focus on the alignment of two networks only. However, in the real-world, people are normally involved in multiple (usually more than two) social networks simultaneously. In this section, we will focus on the simultaneous alignment problem of multiple (more than two) networks, which is called the "multiple anonymized social networks alignment" problem formally [98]. To help illustrate the multi-network alignment problem more clearly, we also give an example in Figure 1, which involves 3 different social networks (i.e., networks I, II and III). Users in these 3 networks are all anonymized and their names are replaced with randomly generated identifiers. Each pair of these 3 anonymized networks can actually share some common users, e.g., "David" participates in both networks I and II simultaneously, "Bob" is using networks I and III concurrently, and "Charles" is involved in all these 3 networks at the same time. Besides these shared anchor users, in these 3 partially aligned networks, some users are involved in one single network only (i.e., the non-anchor users [104]), e.g., "Alice" in network I, "Eva" in network II and "Frank" in network III. The problem studied in this part aims at discovering the anchor links (i.e., the dashed bi-directional red lines) connecting anchor users across these 3 social networks respectively.

The significant difference of the studied problem from existing *two* network alignment problems is due to the "*transitivity law*" that anchor links follow. In traditional set theory, a relation $\mathcal{R}$ is defined to be a *transitive relation* in domain $\mathcal{X}$ iff $\forall a, b, c \in \mathcal{X}, (a,b) \in \mathcal{R} \wedge (b,c) \in \mathcal{R} \rightarrow (a,c) \in \mathcal{R}$. If we treat the union of user account sets of all these social networks as the target domain $\mathcal{X}$ and treat anchor links as the relation $\mathcal{R}$, then anchor links depict a "*transitive relation*" among users across networks. We can take the networks shown in Figure 1 as an example. Let u be a user involved in networks I, II and III simultaneously, whose accounts in these networks are u^I, u^{II} and u^{III} respectively. If anchor links (u^I, u^{II}) and (u^{II}, u^{III}) are identified in aligning networks (I, II) and networks (II, III) respectively (i.e., u^I, u^{II} and u^{III} are discovered to be the same user), then anchor link (u^I, u^{III}) should also exist in the alignment result of networks (I, III) as well. In the multi-network alignment problem, we need to guarantee the inferred anchor links can meet the *transitivity law*. Formally, the multi-network alignment problem can be represented as follows.

Given the n isolated social networks $\{G^{(1)}, G^{(2)}, \cdots, G^{(n)}\}$, the multi-network alignment problem aims at discovering the anchor links among these n networks, i.e., the anchor link sets $\mathcal{A}^{(1,2)}$, $\mathcal{A}^{(1,3)}$, $\cdots$, $\mathcal{A}^{(n-1,n)}$. These n social etworks $G^{(1)}, G^{(2)}, \cdots, G^{(n)}$ are partially aligned and the constraint on anchor links in $\mathcal{A}^{(1,2)}$, $\mathcal{A}^{(1,3)}$, $\cdots$, $\mathcal{A}^{(n-1,n)}$ is *one-to-one*, which also follow the *transitivity law*.

To solve the multi-network alignment problem, two global network alignment methods IsoRankN [38] and UMA (Unsupervised Multi-network Alignment) [98] will be introduced as follows.

4.2.1 IsoRankN

IsoRankN [38] algorithm is an extension to IsoRank. Based on the learning results of IsoRank, IsoRankN further adopts the spectral clustering method on the induced graph of pairwise alignment scores to achieve the final alignment results. The new approach provides significant advantages not only over the original IsoRank but also over other methods. IsoRankN has 4 main steps: (1) initial network alignment with IsoRank, (2) star spread, (3) spectral partition, and (4) star merging, where steps (3) and (4) will repeat until all the nodes are assigned to a cluster.

Initial Network Alignment: Given k isolated networks $G^{(1)}$, $G^{(2)}$, $\cdots$, $G^{(k)}$, IsoRankN computes the local alignment scores of node pairs across networks with IsoRank algorithm. For instance, if the networks are unweighted, the alignment score between nodes $u_l^{(i)}$ and $u_m^{(j)}$ between networks $G^{(i)}$, $G^{(j)}$ can be denoted as.

$$r(u_i^{(1)}, u_j^{(2)}) \quad (16)$$
$$= \sum_{u_m^{(1)} \in \Gamma(u_i^{(1)})} \sum_{u_n^{(2)} \in \Gamma(u_i^{(2)})} \frac{1}{|\Gamma(u_i^{(1)})||\Gamma(u_j^{(2)})|} r(u_m^{(1)}, u_n^{(2)}), \quad (17)$$

It will lead to a weighted k-partite graph, where the links denotes the anchor links across networks weighted by the scores calculated above. If the networks $G^{(1)}, \cdots G^{(k)}$ are all complete graphs, the alignment results will be the maximum weighted cliques. However, in the real world, such an assumption can hardly met, and IsoRankN proposes to use "Star Spread" technique to select a subgraph with high weights.

Star Spread: For each node in a network, e.g., $u_l^{(i)}$ in network $G^{(i)}$, the set of nodes connected with $u_l^{(i)}$ via potential anchor links can be denoted as set $\Gamma(u_l^{(i)})$. The nodes in $\Gamma(u_l^{(i)})$ can be further pruned by removing the nodes connected with *weak* anchor links. Here, the "weak" denotes the anchor links with a low score calculated with IsoRank. Formally, among all the nodes in $\Gamma(u_l^{(i)})$, we can denote the node connected to $u_l^{(i)}$ with the strongest link as

$v^* = \arg_{v \in \Gamma(u_l^{(i)})} \max r(u_l^{(i)}, v)$. For all the nodes with weights lower than $\beta \cdot r(u_l^{(i)}, v^*)$ will be removed from $\Gamma(u_l^{(i)})$ (where β is a threshold parameter), and the remaining nodes together with $u_l^{(i)}$ will form a star structured graph $S_{u_l^{(i)}}$.

Spectral Partition: For each node $u_l^{(i)}$, IsoRankN aims at selecting a subgraph $S^*_{u_l^{(i)}}$ from $S_{u_l^{(i)}}$, which contains the highly weighted neighbors of $u_l^{(i)}$. To achieve such a objective, IsoRankN proposes to identify a subgraph with low *conductance* from $S_{u_l^{(i)}}$ instead. Formally, given a network $G = (\mathcal{V}, \mathcal{E})$, let $\mathcal{S} \subset \mathcal{V}$ denote a subset of G. The *conductance* of the subgraph involving $\mathcal{S}$ can be represented as

$$\phi(\mathcal{S}) = \frac{\sum_{u \in \mathcal{S}} \sum_{v \in \bar{\mathcal{S}}} w_{u,v}}{\min(\mathrm{vol}(\mathcal{S}), \mathrm{vol}(\bar{\mathcal{S}}))}, \tag{18}$$

where $\bar{\mathcal{S}} = \mathcal{V} \setminus \mathcal{S}$, and $\mathrm{vol}(\mathcal{S}) = \sum_{u \in \mathcal{S}} \sum_{v \in \mathcal{V}} w_{u,v}$. IsoRankN points out that a node subset $\mathcal{S}$ containing node $u_l^{(i)}$ can be computed effectively and efficiently with the personalized PageRank algorithm starting from node $u_l^{(i)}$.

Star Merging: Considering that links in the star graph $S^*_{u_l^{(i)}}$ are all the anchor links across networks, there exist no intra-network links at all in $S^*_{u_l^{(i)}}$, e.g., the links in network $G^{(i)}$ only. However, in many cases, there may exist multiple nodes corresponding to the same entity inside the network as well. To solve such a problem, IsoRankN proposes a star merging step to combine several star graphs together, e.g., $S^*_{u_l^{(i)}}$ and $S^*_{u_m^{(j)}}$. Formally, given two star graphs $S^*_{u_l^{(i)}}$ and $S^*_{u_m^{(j)}}$, if the following conditions both hold, $S^*_{u_l^{(i)}}$ and $S^*_{u_m^{(j)}}$ can be merged into one star graph.

$$\forall v \in S^*_{u_m^{(j)}} \setminus \{u_m^{(j)}\}, r(v, u_l^{(i)}) \geq \beta \cdot \max_{v' \in \Gamma(u_l^{(i)})} r(v', u_l^{(i)}), \tag{19}$$

$$\forall v \in S^*_{u_l^{(i)}} \setminus \{u_l^{(i)}\}, r(v, u_m^{(j)}) \geq \beta \cdot \max_{v' \in \Gamma(u_m^{(j)})} r(v', u_m^{(j)}). \tag{20}$$

4.2.2 UMA

The UMA model proposed in [98] addresses the multi-network alignment problem with two steps: (1) unsupervised transitive anchor link inference across multi-networks, and (2) transitive multi-network matching to maintain the *one-to-one constraint*, where the first step is very similar to the matrix inference based alignment algorithm introduced in Section 4.1.3. Next, we will mainly focus on introducing the *transitivity* property on the alignment results.

The *transitivity* property should holds for the alignment of any n networks, where the minimum of n is 3. To help illustrate the *transitivity property* more clearly, here we will use 3 network alignment as an example to introduce the multi-network alignment problem and the UMA model, which can be easily generalized to the case of n networks alignment. Let $G^{(i)}$, $G^{(j)}$ and $G^{(k)}$ be 3 social networks to be aligned concurrently. To accommodate the alignment results and preserve the *transitivity* property, UMA introduces the following *alignment transitivity penalty*:

DEFINITION 15. *(Alignment Transitivity Penalty): Formally, let* $\mathbf{T}^{(i,j)}$, $\mathbf{T}^{(j,k)}$ *and* $\mathbf{T}^{(i,k)}$ *be the inferred binary transitional matrices from* $G^{(i)}$ *to* $G^{(j)}$*, from* $G^{(j)}$ *to* $G^{(k)}$ *and from* $G^{(i)}$ *to* $G^{(k)}$ *respectively among these 3 networks. Based on the adjacency matrices* $\mathbf{S}^{(i)}$, $\mathbf{S}^{(j)}$ *and* $\mathbf{S}^{(k)}$ *of networks* $G^{(i)}$, $G^{(j)}$ *and* $G^{(k)}$*, the alignment transitivity penalty* $C(\{G^{(i)}, G^{(j)}, G^{(k)}\})$ *introduced by the inferred transitional matrices can be quantified as the number of inconsistent social links being mapped from* $G^{(i)}$ *to* $G^{(k)}$ *via two different alignment paths* $G^{(i)} \to G^{(j)} \to G^{(k)}$ *and* $G^{(i)} \to G^{(k)}$*, i.e.,*

$$C(\{G^{(i)}, G^{(j)}, G^{(k)}\}) = \tag{21}$$

$$\left\| (\mathbf{T}^{(j,k)})^\top (\mathbf{T}^{(i,j)})^\top \mathbf{S}^{(i)} \mathbf{T}^{(i,j)} \mathbf{T}^{(j,k)} - (\mathbf{T}^{(i,k)})^\top \mathbf{S}^{(i)} \mathbf{T}^{(i,k)} \right\|_F^2. \tag{22}$$

Alignment transitivity penalty is a general penalty concept and can be applied to n networks $\{G^{(1)}, G^{(2)}, \cdots, G^{(n)}\}$, $n \geq 3$ as well, which can be defined as the summation of penalty introduced by any three networks in the set, i.e.,

$$C(\{G^{(1)}, G^{(2)}, \cdots, G^{(n)}\}) \tag{23}$$

$$= \sum_{\forall \{G^{(i)}, G^{(j)}, G^{(k)}\} \subset \{G^{(1)}, G^{(2)}, \cdots, G^{(n)}\}} C(\{G^{(i)}, G^{(j)}, G^{(k)}\}). \tag{24}$$

Based on the loss function introduced in Section 4.1.3 and the above *alignment transitivity penalty*, the optimal *binary transitional matrices* $\bar{\mathbf{T}}^{(i,j)}$, $\bar{\mathbf{T}}^{(j,k)}$ and $\bar{\mathbf{T}}^{(k,i)}$ should minimize friendship inconsistency and the *alignment transitivity penalty* at the same time, which is learned by addressing an optimization problem in UMA [98]. The objective function aims at obtaining the *hard* mappings among users across different networks and entries in all these *transitional matrices* are binary, which can lead to a fatal drawback: *hard assignment* can be neither possible nor realistic for networks with star structures as proposed in [35] and the hard subgraph isomorphism [36] is NP-hard. To address the function, UMA proposes to relax the hard binary constraints on the variables first and solve the function with gradient descent. Furthermore, based on the learning results UMA keeps the one-to-one constraint on anchor links by selecting those which can maximize the overall existence probabilities while maintaining the *matching transitivity* property at the same time.

5. LINK PREDICTION

Given a screenshot of an online social network, the problem of inferring the missing links or the links to be formed in the future is called the *link prediction* problem [39; 23; 104]. Link prediction problem has concrete applications in the real world, and many social network services can be cast to the link prediction problem. For instance, the friend recommendations problem in online social networks can be modeled as the social link prediction problem among users. Users' trajectory prediction problem can be formulated as the prediction task of potential checkin links between users and offline POIs (point of interest) in location based social networks. The user identifier resolution problem across networks (i.e., the network alignment problem introduced in the previous section) can be modeled as the anchor link prediction problem of user accounts across different online social networks.

In this section, we will introduce the general link prediction problems in online social networks. Formally, given the training set $\mathcal{T}_{train}$ involving links belong to different classes ($\mathcal{Y} = \{+1, -1\}$ denoting the links that have been/will be formed and those will never be formed) and the test set $\mathcal{T}_{test}$ (with unknown labels), the link prediction problem aims at building a mapping $f : \mathcal{T}_{test} \to \mathcal{Y}$ to infer the potential labels of links in the test set $\mathcal{T}_{test}$.

Depending on the scenarios of the link prediction problems, the existing links prediction works can be divided into several different categories. Traditional link prediction problems are mainly focused on inferring the links in one single homogeneous network [87; 104], like inferring the friendship links among users in online social networks or co-author links in bibliographic networks. As the network structures are becoming more and more complicated, many of them are modeled as the heterogeneous networks involving different types of nodes and complex connections among them.

The heterogeneity of the networks leads to many new link prediction problems, like predicting the links between nodes belonging to different categories and the concurrent inference of multiple types of links in the heterogeneous networks [87; 99]. In recent years, many online social networks have appeared, and lots of new research opportunities exist for researchers and practitioners to study the link prediction problem from the cross-network perspective [87; 104; 84; 83].

Meanwhile, depending on the learning settings used in the problem formulation, the existing link prediction works can be categorized in another way. For some of the link prediction models, they calculate the user-pair closeness as the prediction result without needing any training data, which are referred to as the *unsupervised link prediction models* [39]. For some other models, they will label the known links into different classes, and use them as the training set to learn a supervised classification models as the base model instead. These models are called the *supervised link prediction models* [23]. Usually, manual labeling of the links is very expensive and tedious. In recent years, many of the works have proposed to apply semi-supervised learning techniques in the link prediction problem to utilize the links without labels [87; 104; 84; 83].

In this part, we will introduce the link prediction problems in online social networks, including the *traditional homogeneous link prediction* and *cross-network link prediction*.

5.1 Homogeneous Network Link Prediction

Traditional link prediction problems are mainly studied based on one homogeneous network, involving one single type of nodes and links. In this section, we will first briefly introduce how to use the social closeness measures for link prediction tasks. To integrate different social closeness measures together in the link prediction task, we will talk about the supervised link prediction model. Furthermore, several semi-supervised link prediction models will also be introduced in this section, which formulate the link prediction as a semi-supervised learning problem.

5.1.1 Unsupervised Link Prediction

Given a screenshot of a homogeneous network $G = (\mathcal{V}, \mathcal{E})$, the unsupervised link prediction methods [39] aims at inferring the potential links that will be formed in the future. Usually, the unsupervised link prediction models will calculate the closeness scores of the node pairs, which will be used as the predicted confidence scores of these links. Depending on the specific scenario and the link formation assumptions applied, different measures have been proposed for the link prediction models.

Local Neighbor based Predicators: Local neighbor based predicators are based on regional social network information, i.e., neighbors of users in the network. Consider, for example, given a social link (u, v) in network G, where u and v are both users in G, the neighbor sets of u, v can be represented as $\Gamma(u)$ and $\Gamma(v)$ respectively. Based on $\Gamma(u)$ and $\Gamma(v)$, the following predicators measuring the proximity of users u and v in network G can be obtained.

1. *Preferential Attachment Index* (PA) [5]:
$$PA(u, v) = |\Gamma(u)| \, |\Gamma(v)| \,. \tag{25}$$
$PA(u, v)$ uses the product of the degrees of users u and v in the network as the proximity measure, considering that new links are more likely to appear between users who have large number of social connections.

2. *Common Neighbor* (CN) [24]:
$$CN(u, v) = |\Gamma(u) \cap \Gamma(v)| \,. \tag{26}$$
$CN(u, v)$ uses the number of shared neighbor as the proximity score of user u and v. The larger $CN(u, v)$ is, the closer user u and v are in the network.

3. *Jaccard's Coefficient* (JC) [24]:
$$JC(u, v) = \frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|}. \tag{27}$$
$JC(u, v)$ takes the total number of neighbors of u and v into account, considering that $CN(u, v)$ can be very large because each one has a lot of neighbors rather than they are strongly related to each other.

4. *Adamic/Adar Index* (AA) [1]:
$$AA(u, v) = \sum_{w \in (\Gamma(u) \cap \Gamma(v))} \frac{1}{\log |\Gamma(w)|}. \tag{28}$$
Different from $JC(u, v)$, $AA(u, v)$ further gives each common neighbor of user u and v a weight, $\frac{1}{\log|\Gamma(w)|}$, to denote its importance.

5. *Resource Allocation Index* (RA) [107]:
$$RA(u, v) = \sum_{w \in (\Gamma(u) \cap \Gamma(v))} \frac{1}{|\Gamma(w)|}. \tag{29}$$
$RA(u, v)$ gives each common neighbor a weight $\frac{1}{|\Gamma(w)|}$ to represent its importance, where those with larger degrees will have a less weight number.

All these predicators are called *local neighbor based predicators* as they are all based on users' local social network information.

Global Path based Predicators: In addition to the local neighbor based predicators, many other predicators based on paths in the network have also been proposed to measure the proximity among users.

1. *Shortest Path* (SP) [23]:
$$SP(u, v) = \min\{|p_{u \rightsquigarrow v}|\}, \tag{30}$$
where $p_{u \rightsquigarrow v}$ denotes a path from u to v in the network and $|p|$ represents the length of path p.

2. *Katz* [28]:
$$Katz(u, v) = \sum_{l=1}^{\infty} \beta^l \left| p^l_{u \rightsquigarrow v} \right|, \tag{31}$$
where $p^l_{u \rightsquigarrow v}$ is the set of paths of length l from u to v and parameter $\beta \in [0, 1]$ is a regularizer of the predicator. Normally, a small β favors shorter paths as β^l can decay very quickly when β is small, in which case $Katz(u, v)$ will be behave like the predicators based on local neighbors.

Random Walk based Link Prediction: In addition to the unsupervised link predicators which can be obtained from the networks directly, there exists another category link prediction methods which can calculate the proximity scores among users based on *random walk* [21; 19; 33; 4; 68; 42; 24]. In this part, we will introduce the concept of random walk at first. Next, we will introduce the proximity measures based on random walk, which include the *commute time* [19; 42; 24], *hitting time* [19; 42; 24] and *cosine similarity* [19; 42; 24].

Let matrix $\mathbf{A}$ be the adjacency matrix of network G, where $A(i, j) = 1$ iff social link $(u_i, u_j) \in \mathcal{E}$, where $u_i, u_j \in \mathcal{V}$. The normalized matrix of $\mathbf{A}$ by rows will be $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$, where diagonal matrix $\mathbf{D}$ of $\mathbf{A}$ has value $D(i, i) = \sum_j A(i, j)$ on its diagonal and $P(i, j)$ stores the probability of stepping on node $u_j \in \mathcal{V}$ from node $u_i \in \mathcal{V}$. Let entries in vector $\mathbf{x}^{(\tau)}(i)$ denote the probabilities that a random walker is at user node $u_i \in \mathcal{V}$ at time τ. Then we have the updating equation of entry $\mathbf{x}^{(\tau)}(i)$ via the random walk as follows:

$$\mathbf{x}^{(\tau+1)}(i) = \sum_j \mathbf{x}^{(\tau)}(j)\mathbf{P}(j,i). \tag{32}$$

In other words, the updating equation of vector $\mathbf{x}$ will be represented as:

$$\mathbf{x}^{(\tau+1)} = \mathbf{P}\mathbf{x}^{(\tau)}. \tag{33}$$

By keeping updating $\mathbf{x}$ according to the following equation until convergence, we can have the stationary vector $\mathbf{x}^{(\tau+1)}$ as

$$\begin{cases} \mathbf{x}^{(\tau+1)} = \mathbf{P}^T\mathbf{x}^{(\tau)}, \\ \mathbf{x}^{(\tau+1)} = \mathbf{x}^{(\tau)}. \end{cases} \tag{34}$$

The above equation is equivalent to

$$\mathbf{v} = \mathbf{P}^T\mathbf{v}, \tag{35}$$

where $\mathbf{v}$ denotes the stationary random walk probability vector.
The above equation denotes that the final stationary distribution vector $\mathbf{v}$ is actually a eigenvector of matrix $\mathbf{P}^T$ corresponding to eigenvalue 1. Some existing works have pointed out that if a markov chain is *irreducible* [19] and *aperiodic* [19] then the largest eigenvalue of the transition matrix will be equal to 1 and all the other eigenvalues will be strictly less than 1. In addition, in such a condition, there will exist one single unique stationary distribution which is vector $\mathbf{v}$ obtained at convergence of the updating equations.

Proximity Measures based on Random Walk

1. *Hitting Time* (HT):

$$HT(u,v) = \mathbb{E}\left(\min\{\tau | \tau \in \mathbb{N}^+, X^{(\tau)} = v \wedge X^0 = u\}\right), \tag{36}$$

where variable $X^{(\tau)} = v$ denotes that a random walker is at node v at time τ.

$HT(u,v)$ counts the average steps that a random walker takes to reach node v from node u. According to the definition, the hitting time measure is usually asymmetric, $HT(u,v) \neq HT(v,u)$. Based on matrix $\mathbf{P}$ defined before, the definition of $HT(u,v)$ can be redefined as [19]:

$$HT(u,v) = 1 + \sum_{w \in \Gamma(u)} P_{u,w} HT(w,v). \tag{37}$$

2. *Commute Time* (CT):

$$CT(u,v) = HT(u,v) + HT(v,u). \tag{38}$$

$CT(u,v)$ counts the expectation of steps used to reach node u from v and those needed to reach node v from u. According to existing works, the commute time, $CT(u,v)$, can be obtained as follows

$$CT(u,v) = 2m(L^\dagger_{u,u} + L^\dagger_{v,v} - 2L^\dagger_{u,v}), \tag{39}$$

where $\mathbf{L}^\dagger$ is the pseudo-inverse of matrix $\mathbf{L} = \mathbf{D}_A - \mathbf{A}$.

3. *Cosine Similarity based on* $\mathbf{L}^\dagger$ (CS):

$$CS(u,v) = \frac{\mathbf{x}_u^T\mathbf{x}_v}{\sqrt{(\mathbf{x}_u^T\mathbf{x}_u)(\mathbf{x}_v^T\mathbf{x}_v)}}, \tag{40}$$

where, $\mathbf{x}_u = (\mathbf{L}^\dagger)^{\frac{1}{2}}\mathbf{e}_u$ and vector $\mathbf{e}_u$ is a vector of 0s except the entries corresponding to node u that is filled with 1. According to existing works [19; 42], the cosine similarity based on $\mathbf{L}^\dagger$, $CS(u,v)$, can be obtained as follows,

$$CS(u,v) = \frac{L^\dagger_{u,v}}{\sqrt{L^\dagger_{u,u}L^\dagger_{v,v}}}. \tag{41}$$

4. *Random Walk with Restart* (RWR): Based on the definition of random walk, if the walker is allowed to return to the starting point with a probability of $1-c$, where $c \in [0,1]$, then the new random walk method is formally defined as *random walk with restart*, whose updating equation is shown as follows:

$$\begin{cases} \mathbf{x}_u^{(\tau+1)} = c\mathbf{P}^T\mathbf{x}_u^{(\tau)} + (1-c)\mathbf{e}_u, \\ \mathbf{x}_u^{(\tau+1)} = \mathbf{x}_u^{(\tau)}. \end{cases} \tag{42}$$

Keep updating $\mathbf{x}$ until convergence, the stationary distribution vector $\mathbf{x}$ can meet

$$\mathbf{x}_u = (1-c)(\mathbf{I} - c\mathbf{P}^T)^{-1}\mathbf{e}_u. \tag{43}$$

The proximity measure based on random walk with restart between user u and v will be

$$RWR(u,v) = \mathbf{x}_u(v), \tag{44}$$

where $\mathbf{x}_u(v)$ denotes the entry corresponding to v in $\mathbf{x}_u$.

5.1.2 Supervised Link Prediction

In some cases, links in the networks are explicitly categorized into different groups, like links denoting friends vs those representing enemies, friends (formed connections) vs strangers (no connections). Given a set of labeled links, e.g., set $\mathcal{E}$, containing links belonging to different classes, the *supervised link prediction* [23] problem aims at building a supervised learning model with the labeled set. The learnt model will be applied to determine the labels of links in the test set. In this part, we still take the link formation problem as an example to illustrate the supervised link prediction model.
To represent each of the social links, like link $l = (u,v) \in \mathcal{E}$ between nodes u and v, a set of features representing the characteristics of the link l or nodes u, v will be extracted in the model building. Normally, the features can be extracted for links in the prediction task can be divided into two categories:

Link Feature Extraction

- *Features of Nodes*: The characteristics of the nodes can be denoted by various measures, like these various node centrality measures. For instance, for the link (u,v), based on the known links in the training set, the centrality measures can be computed based on degree, normalized degree, eigenvector, Katz, PageRank, Betweenness of nodes u and v as part of the features for link (u,v).
- *Features of Links*: The characteristics of the links in the networks can be calculated by computing the closeness between the nodes composing the nodes. For instance, for link (u,v), based on the known links in the training set, the closeness measures can be computed based on reciprocity, common neighbor, Jaccard's coefficient, Adamic/Adar, shortest path, Katz, hitting time, commute time, etc. between nodes u and v as the features for link (u,v).

We can append the features for nodes u, v and those for link (u,v) together and represent the extracted feature vector for link $l = (u,v)$ as vector $\mathbf{x}_l \in \mathbb{R}^{k \times 1}$, whose length is k in total.

Link Prediction Model

With the training set $\mathcal{L}_{train}$, the feature vectors and labels for the links in $\mathcal{L}_{train}$ can be represented as the training data $\{(\mathbf{x}_l, y_l)\}_{l \in \mathcal{L}_{train}}$. Meanwhile, with the testing set $\mathcal{L}_{test}$, the features extracted for the links in it can be represented as $\{\mathbf{x}_l\}_{l \in \mathcal{L}_{train}}$. Different classification models can be used as the base model for the link prediction task, like the Decision Tree, Artificial Neural Network and Support Vector Machine (SVM) [2]. The model can be trained with the

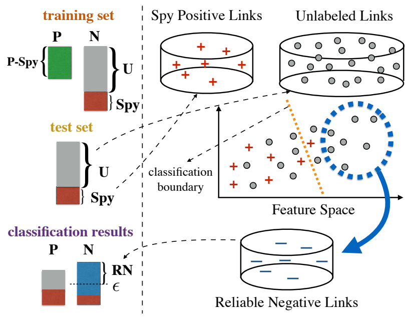

Figure 2: PU Link Prediction.

training data, and the labels of links in the test can be determined by applying models to the test set.

Depending on the specific model being applied, the output of the link prediction result can include (1) the predicted labels of the links, and (2) the prediction confidence scores/probability scores of links in the test set.

5.1.3 PU Link Prediction

In the real world, for the links which are unlabeled, some of them can actually be formed in the future. In this subsection, we will introduce a method MLI to solve the *PU link prediction* problem in one single network [104]. From a given network, e.g., G, two disjoint sets of links: connected (i.e., formed) links $\mathcal{P}$ and unconnected links $\mathcal{U}$, can be obtained. To differentiate these links, MLI uses a new concept "*connection state*", z, to show whether a link is connected (i.e., formed) or unconnected in network G. For a given link l, if l is connected in the network, then $z(l) = +1$; otherwise, $z(l) = -1$. As a result, MLI can have the "*connection states*" of links in $\mathcal{P}$ and $\mathcal{U}$ to be: $z(\mathcal{P}) = \mathbf{+1}$ and $z(\mathcal{U}) = \mathbf{-1}$.

Besides the "*connection state*", links in the network can also have their own "*labels*", y, which can represent whether a link is to be formed or will never be formed in the network. For a given link l, if l has been formed or to be formed, then $y(l) = +1$; otherwise, $y(l) = -1$. Similarly, MLI can have the "*labels*" of links in $\mathcal{P}$ and $\mathcal{U}$ to be: $y(\mathcal{P}) = \mathbf{+1}$ but $y(\mathcal{U})$ can be either $+1$ or -1, as $\mathcal{U}$ can contain both links to be formed and links that will never be formed.

By using $\mathcal{P}$ and $\mathcal{U}$ as the positive and negative training sets, MLI can build a *link connection prediction model* $\mathcal{M}_c$, which can be applied to predict whether a link exists in the original network, i.e., the *connection state* of a link. Let l be a link to be predicted, by applying $\mathcal{M}_c$ to classify l, the *connection probability* of l can be represented to be:

DEFINITION 16. *(Connection Probability): The probability that link l's connection states is predicted to be connected (i.e., $z(l) = +1$) is formally defined as the connection probability of link l: $p(z(l) = +1|\mathbf{x}(l))$, where $\mathbf{x}(l)$ denotes the feature vector extracted for link l based on meta path.*

Meanwhile, if we can obtain a set of links that "will never be formed", i.e., "-1" links, from the network, which together with $\mathcal{P}$ ("+1" links) can be used to build a *link formation prediction model*, $\mathcal{M}_f$, which can be used to get the *formation probability* of l to be:

DEFINITION 17. *(Formation Probability): The probability that link l's label is predicted to be formed or will be formed (i.e., $y(l) = +1$) is formally defined as the formation probability of link l: $p(y(l) = +1|\mathbf{x}(l))$.*

However, from the network, we have no information about "links that will never be formed" (i.e., "-1" links). As a result, the *formation probabilities* of potential links that we aim to obtain can be very challenging to calculate. Meanwhile, the correlation between link l's *connection probability* and *formation probability* has been proved in existing works [15] to be:

$$p(y(l) = +1|\mathbf{x}(l)) \propto p(z(l) = +1|\mathbf{x}(l)). \tag{45}$$

In other words, for links whose *connection probabilities* are low, their *formation probabilities* will be relatively low as well. This rule can be utilized to extract links which can be more likely to be the reliable "-1" links from the network. MLI proposes to apply the the *link connection prediction model* $\mathcal{M}_c$ built with $\mathcal{P}$ and $\mathcal{U}$ to classify links in $\mathcal{U}$ to extract the *reliable negative link set*.

DEFINITION 18. *(Reliable Negative Link Set): The reliable negative links in the unconnected link set $\mathcal{U}$ are those whose connection probabilities predicted by the link connection prediction model, $\mathcal{M}_c$, are lower than threshold $\epsilon \in [0, 1]$:*

$$\mathcal{RN} = \{l | l \in \mathcal{U}, p(z(l) = +1|\mathbf{x}(l)) < \epsilon\}. \tag{46}$$

Some Heuristic methods have been proposed to set the optimal threshold ϵ, e.g., the *spy technique* proposed in [41]. As shown in Figure 2, MLI proposes randomly select a subset of links in $\mathcal{P}$ as the spy, $\mathcal{SP}$, whose proportion is controlled by $s\%$. $s\% = 15\%$ is used as the default sample rate in [104]. Sets $(\mathcal{P} - \mathcal{SP})$ and $(\mathcal{U} \cup \mathcal{SP})$ are used as positive and negative training sets to the *spy prediction* model, $\mathcal{M}_s$. By applying $\mathcal{M}_s$ to classify links in $(\mathcal{U} \cup \mathcal{SP})$, their *connection probabilities* can be represented to be:

$$p(z(l) = +1|\mathbf{x}(l)), l \in (\mathcal{U} \cup \mathcal{SP}), \tag{47}$$

and parameter ϵ is set as the minimal *connection probability* of spy links in $\mathcal{SP}$:

$$\epsilon = \min_{l \in \mathcal{SP}} p(z(l) = +1|\mathbf{x}(l)). \tag{48}$$

With the extracted *reliable negative link set* $\mathcal{RN}$, MLI can solve the *PU link prediction* problem with *classification based link prediction methods*, where $\mathcal{P}$ and $\mathcal{RN}$ are used as the positive and negative training sets respectively.

5.2 Inter-Network Link Prediction

Besides the link prediction problems in one single target network, some research works have been done on simultaneous link prediction in multiple aligned online social networks concurrently [87; 104; 84; 83].

5.2.1 MLI

Method MLI proposed in [104] is a general link prediction framework and can be applied to predict social links in n *partially aligned networks* simultaneously. When it comes to n partially aligned network $G^{(1)}, \cdots, G^{(n)}$, the optimal labels of potential links $\{\mathcal{L}^{(1)}, \mathcal{L}^{(2)}, \cdots, \mathcal{L}^{(n)}\}$ of networks $G^{(1)}, \cdots, G^{(n)}$ will be:

$$\hat{\mathcal{Y}}^{(1)}, \hat{\mathcal{Y}}^{(2)}, \cdots, \hat{\mathcal{Y}}^{(n)} = \arg \max_{\mathcal{Y}^{(1)}, \cdots, \mathcal{Y}^{(n)}} \tag{49}$$

$$p\Big(y(\mathcal{L}^{(1)}) = \mathcal{Y}^{(1)}, \cdots, y(\mathcal{L}^{(n)}) = \mathcal{Y}^{n} | G^{(1)}, \cdots, G^{(n)}\Big). \tag{50}$$

The above target function is very complex to solve and, in [104], MLI proposes to obtain the solution by updating one variable, e.g., $\mathcal{Y}^{(1)}$, and fix other variables, e.g., $\mathcal{Y}^{(2)}, \cdots, \mathcal{Y}^{(n)}$, alternatively with the following equation [87]:

$$\begin{cases} (\hat{\mathcal{Y}}^{(1)})^{(\tau)} = \arg\max_{\mathcal{Y}^{(1)}} p\Big(y(\mathcal{L}^{(1)}) = \mathcal{Y}^{(1)} | G^{(1)}, G^{(2)}, \cdots, G^{(n)}, \\ \qquad (\hat{\mathcal{Y}}^{2})^{(\tau-1)}, (\hat{\mathcal{Y}}^{3})^{(\tau-1)}, \cdots, (\hat{\mathcal{Y}}^{n})^{(\tau-1)}\Big), \\ (\hat{\mathcal{Y}}^{(2)})^{(\tau)} = \arg\max_{\mathcal{Y}^{(2)}} p\Big(y(\mathcal{L}^{(2)}) = \mathcal{Y}^{(2)} | G^{(1)}, G^{(2)}, \cdots, G^{(n)}, \\ \qquad (\hat{\mathcal{Y}}^{(1)})^{(\tau)}, (\hat{\mathcal{Y}}^{(3)})^{(\tau-1)}, \cdots, (\hat{\mathcal{Y}}^{(n)})^{(\tau-1)}\Big), \\ \qquad \cdots\cdots \\ (\hat{\mathcal{Y}}^{(n)})^{(\tau)} = \arg\max_{\mathcal{Y}^{(n)}} p\Big(y(\mathcal{L}^{(n)}) = \mathcal{Y}^{(n)} | G^{(1)}, G^{(2)}, \cdots, G^{(n)}, \\ \qquad (\hat{\mathcal{Y}}^{(1)})^{(\tau)}, (\hat{\mathcal{Y}}^{(2)})^{(\tau)}, \cdots, (\hat{\mathcal{Y}}^{(n-1)})^{(\tau)}\Big). \end{cases} \tag{51}$$

When predicting social links in network $G^{(i)}$, MLI can extract features based on the *intra-network social meta path* extracted from $G^{(i)}$ and those extracted based on the *inter-network social meta path* across $G^{(1)}$, $G^{(2)}$, $\cdots$, $G^{(i-1)}$, $G^{(i+1)}$, $\cdots$, $G^{(n)}$ for links in $\mathcal{P}^{(i)}$, $\mathcal{U}^{(i)}$ and $\mathcal{L}^{(i)}$. Feature vectors $\mathbf{x}(\mathcal{P})$, $\mathbf{x}(\mathcal{U})$ as well as the labels, $y(\mathcal{P})$, $y(\mathcal{U})$, of links in $\mathcal{P}$ and $\mathcal{U}$ are passed to the PU link prediction model $\mathcal{M}^{(i)}$ and the meta path selection model $\mathcal{MS}^{(i)}$. The formation probabilities of links in $\mathcal{L}^{(i)}$ predicted by model $\mathcal{M}^{(i)}$ will be used to update the network by replace the weights of $\mathcal{L}^{(i)}$ with the newly predicted formation probabilities. The initial weights of these potential links in $\mathcal{L}^{(i)}$ are set as 0. After finishing these steps on $G^{(i)}$, we will move to conduct similar operations on $G^{(i+1)}$. MLI iteratively predicts links in $G^{(1)}$ to $G^{(n)}$ alternatively in a sequence until the results in all of these networks converge.

5.2.2 *SLAMPRED*

The cross-network link prediction model SLAMPRED introduced in [83] aims at inferring the links for emerging networks based on semi-supervised learning setting. SLAMPRED proposes to embed the feature vectors of links from aligned networks into a shared feature space. Via the shared feature space, knowledge from the source networks will be effectively transferred to the target network.

Formally, let G^t denote the target emerging network, where the user set can be represented as $\mathcal{U}^t$. The existing social connections in G^t can be represented as the binary social adjacency matrix $\mathbf{A}^t \in \{0,1\}^{|\mathcal{U}^t|\times|\mathcal{U}^t|}$, where entry $A^t(i,j) = 1$ iff the corresponding social link (u_i^t, u_j^t) exists between users u_i^t and u_j^t in G^t. In the studied problem here, our objective is to infer the potential unobserved social links for the target network, which can be achieved by finding a sparse and low-rank predictor matrix $\mathbf{S} \in \mathcal{S}$ from some convex admissible set $\mathcal{S} \subset \mathbb{R}^{|\mathcal{U}^t|\times|\mathcal{U}^t|}$. Meanwhile, the inconsistency between the inferred matrix $\mathbf{S}$ and the observed social adjacency matrix $\mathbf{A}^t$ can be represented as the loss function $l(\mathbf{S}, \mathbf{A}^t)$. The optimal social link predictor for the target network can be achieved by minimizing the loss term, i.e.,

$$\arg\min_{\mathbf{S}\in\mathcal{S}} l(\mathbf{S}, \mathbf{A}^t). \tag{52}$$

Meanwhile, to utilize the other heterogeneous information available in the emerging network G^t and other external source networks $G^1, G^2, \cdots, G^K$. SLAMPRED proposes to extract a group of intimacy features and project the link instances to a shared feature space as introduced in [83]. The adapted features from the target network and external sources can be represented as tensors $\hat{\mathbf{X}}^t, \hat{\mathbf{X}}^1, \cdots, \hat{\mathbf{X}}^K$. Formally, the intimacy scores of the potential social links based on these adapted features from the external source networks can be represented as

$$int(\mathbf{S}, \hat{\mathbf{X}}^i) = \sum_{k=1}^{d^t} \left\| \mathbf{S} \circ \hat{\mathbf{X}}^i(k,:,:) \right\|_1, i \in \{t, 1, 2, \cdots, K\} \tag{53}$$

$$int(\mathbf{S}, \hat{\mathbf{X}}^1, \cdots, \hat{\mathbf{X}}^K) = \sum_{k=1}^{K} \alpha^i \cdot int(\mathbf{S}, \hat{\mathbf{X}}^k), \tag{54}$$

where users in $\hat{\mathbf{X}}^k$ are organized in the same order as $\mathbf{X}^t$. Parameters α^i denotes the importance of the information transferred from the source network G^i.

By adding the intimacy terms about the source networks into the objective function, the equation can be rewriten as follows:

$$\arg\min_{\mathbf{S}\in\mathcal{S}} \; l(\mathbf{S}, \mathbf{A}^t) - \alpha^t \cdot int(\mathbf{S}, \hat{\mathbf{X}}^t) - \sum_{k=1}^{K} \alpha^i \cdot int(\mathbf{S}, \hat{\mathbf{X}}^k)) \tag{55}$$

$$+ \gamma \cdot \|\mathbf{S}\|_1 + \tau \cdot \|\mathbf{S}\|_*, \tag{56}$$

where $\|\mathbf{S}\|_1$ and $\|\mathbf{S}\|_*$ denote the L_1-norm and trace-norm of matrix $\mathbf{S}$ respectively.

By studying the objective function, we observe that the intimacy terms are convex while the empirical loss term $l(\mathbf{S}, \mathbf{A}^t)$ is non-convex. In [83], the introduced model proposes to approximate it with other classical loss functions (e.g., the hinge loss and the Frobenius norm) instead, and the convex squared Frobenius norm loss function is used in [83] (i.e., $l(\mathbf{S}, \mathbf{A}^t) = \left\|\mathbf{S} - \mathbf{A}^t\right\|_F^2$). Therefore, the above objective function can be represented as a convex loss term minus another convex term together with two convex non-differentiable regularizers, which actually renders the objective function non-trivial. According to the existing works [75; 64], this kind of objective function can be addressed with the concave-convex procedure (CCCP). CCCP is a majorization-minimization algorithm that solves the difference of convex functions problems as a sequence of convex problems. Meanwhile, the regularization terms can be effectively handled with the proximal operators in each iteration of the CCCP process.

6. COMMUNITY DETECTION

In the real-world online social networks, users tend to form different social groups [3]. Users belonging to the same groups usually have more frequent interactions with each other, while those in different groups will have less interactions on the other hand [106]. Formally, such social groups form by users in online social networks are called the online social communities [97]. Online social communities will partition the network into a number of connected components, where the intra-community social connections are usually far more dense compared with the inter-community social connections [97]. Meanwhile, from the mathematical representation perspective, due to these online social communities, the social network adjacency matrix tend to be not only sparse but also low-rank [101].

Identifying the social communities formed by users in online social networks is formally defined as the *community detection* problem [97; 95; 26]. Community detection is a very important problem for online social network studies, as it can be crucial prerequisite for numerous concrete social network services: (1) better organization of users' friends in online social networks (e.g., Facebook and Twitter), which can be achieved by applying community detection techniques to partition users' friends into different categories, e.g., schoolmates, family, celebrities, etc. [16]; (2) better recommender systems for users with common shopping preference in e-commerce social sites (e.g., Amazon and Epinions), which can be addressed by grouping users with similar purchase records into the same clusters prior to recommender system building [56]; and (3) better identification of influential users [69] for advertising campaigns in online social networks, which can be attained by selecting the most influential users in each community as the seed users in the viral marketing [55].

In this section, we will focus on introducing the *social community detection* problem in online social networks. Given a heterogeneous network G with node set $\mathcal{V}$, the involved user nodes in network G can be represented as set $\mathcal{U} \subset \mathcal{V}$. Based on both the social structures among users as well as the diverse attribute information from the network G, the *social community detection* problem aims at partitioning the user set $\mathcal{U}$ into several subsets $\mathcal{C} = \{\mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_k\}$, where each subset $\mathcal{U}_i, i \in \{1, 2, \cdots, k\}$ is called a social community. Term k formally denotes the total number of partitioned communities, which is usually provided as a hyper-parameter in the problem.

Depending on whether the users are allowed to be partitioned into multiple communities simultaneously or not, the *social community* *detection* problem can actually be categorized into two different types:

- *Hard Social Community Detection*: In the *hard social community detection* problem, each user will be partitioned into one single community, and all the social communities are disjoint without any overlap. In other words, given the communities $\mathcal{C} = \{\mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_k\}$ detected from network G, we have $\mathcal{U} = \bigcup_i \mathcal{U}_i$ and $\mathcal{U}_i \cap \mathcal{U}_j = \emptyset, \forall i, j \in \{1, \cdots, k\}$.
- *Soft Social Community Detection*: In the *soft social community detection* problem, users can belong to multiple social communities simultaneously. For instance, if we apply the *Mixture-of-Gaussian Soft Clustering* algorithm as the base community detection model [105; 74], each user can belong to multiple communities with certain probabilities. In the *soft social community detection* result, the communities are no longer disjoint and will share some common users with other communities.

Meanwhile, depending on the network connection structures, the *community detection* problem can be categorized as *directed network community detection* [43] and *undirected network community detection* [106]. Based on the heterogeneity of the network information, the *community detection* problem can be divided into the *homogeneous network community detection* [72] and *heterogeneous network community detection* [57; 66; 85; 101]. Furthermore, according to the number of networks involved, the *community detection* problem involves *single network community detection* [37] and *multiple network community detection* [97; 95; 26; 85; 101]. In this section, we will take the *hard community detection problem* as an example to introduce the existing models proposed for conventional (one single) *homogeneous social network*, and especially the recent broad learning based (multiple aligned) *heterogeneous social networks* [32; 86; 87; 104] respectively.

This section is organized as follows. At the beginning, in Section 6.1, we will introduce the community detection problem and the existing methods proposed for traditional one single homogeneous networks. After that, we will talk about the latest research works on social community detection across multiple aligned heterogeneous networks. In Section 6.2, we will be focused on the concurrent mutual community detection [97] across multiple aligned heterogeneous networks simultaneously, where information from other aligned networks will be applied to refine their community detection results mutually.

6.1 Homogeneous Network Community Detection

Social community detection problem has been studied for a long time, and many community detection models have been proposed based on different types of techniques. In this section, we will talk about the social community detection problem for one single homogeneous network G, whose objective is to partition the user set $\mathcal{U}$ in network G into k disjoint subsets $\mathcal{C} = \{\mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_k\}$, where $\mathcal{U} = \bigcup_i \mathcal{U}_i$ and $\mathcal{U}_i \cap \mathcal{U}_j = \emptyset, \forall i, j \in \{1, 2, \cdots, k\}$. Several different community detection methods will be introduced, which include *node proximity based community detection*, *modularity maximization based community detection*, and *spectral clustering based community detection*.

6.1.1 Node Proximity based Community Detection

The *node proximity based community detection* method assumes that "close nodes tend to be in the same communities, while the nodes far away from each other will belong to different communities". Therefore, the *node proximity based community detection* model partition the nodes into different clusters based on the node proximity measures [39]. Various node proximity measures can be used here, including the node *structural equivalence* to be introduced as follows, as well as various node closeness measures as introduced in Section *5.1.1*.

In a homogeneous network G, the proximity of nodes, like u and v, can be calculated based on their positions and connections in the network structure.

DEFINITION 19. *(Structural Equivalence): Given a network $G = (\mathcal{V}, \mathcal{E})$, two nodes $u, v \in \mathcal{V}$ are said to be structural equivalent iff*

1. *Nodes u and v are not connected and u and v share the same set of neighbors (i.e., $(u, v) \notin \mathcal{E} \wedge \Gamma(u) = \Gamma(v)$),*
2. *Or u and v are connected and excluding themselves, u and v share the same set of neighbors (i.e., $(u, v) \in \mathcal{E} \wedge \Gamma(u) \setminus \{v\} = \Gamma(v) \setminus \{u\}$).*

For the nodes which are *structural equivalent*, they are *substitutable* and switching their positions will not change the overall network structure. The *structural equivalence* concept can be applied to partition the nodes into different communities. For the nodes which are *structural equivalent*, they can be grouped into the same communities, while for the nodes which are not equivalent in their positions, they will be partitioned into different groups. However, the *structural equivalence* can be too restricted for practical application in detecting the communities in real-world social networks. Computing the *structural equivalence* relationships among all the node pairs in the network can lead to very high time cost. What's more, the *structural equivalence* relationship will partition the social network structure into lots of small-sized fragments, since the users will have different social patterns in making friends online and few user will have identical neighbors actually.

To avoid the weakness mentioned above, some other measures are proposed to measure the proximity among nodes in the networks. For instance, as introduced in Section *5.1.1*, the node closeness measures based on the social connections can all be applied here to compute the node proximity, e.g., "common neighbor", "Jaccard's coefficient". Here, if we use "common neighbor" as the proximity measure, by applying the "common neighbor" measure to the network G, the network G can be transformed into a set of instances $\mathcal{V}$ with mutual closeness scores $\{c(u, v)\}_{u,v \in \mathcal{V}}$. Some existing similarity/distance based clustering algorithms, like k-Medoids, can be applied to partition the users into different communities.

6.1.2 Modularity based Community Detection

Besides the pairwise proximity of nodes in the network, the connection strength of a community is also very important in the community detection process. Different measures have been proposed to compute the strength of a community, like the *modularity* measure [48] to be introduced in this part.

The *modularity* measure takes account of the node degree distribution. For instance, given the network G, the expected number of links existing between nodes u and v with degrees $D(u)$ and $D(v)$ can be represented as $\frac{D(u) \cdot D(v)}{2|\mathcal{E}|}$. Meanwhile, in the network, the real number of links existing between u and v can be denoted as entry $A[u, v]$ in the social adjacency matrix $\mathbf{A}$. For the user pair (u, v) with a low expected connection confidence score, if they are connected in the real world, it indicates that u and v have a relatively strong relationship with each other. Meanwhile, if the community detection algorithm can partition such user pairs into the same group, it will be able to identify very strong social communities from the network.

Based on such an intuition, the strength of a community, e.g., $\mathcal{U}_i \in \mathcal{C}$ can be defined as

$$\sum_{u,v \in \mathcal{U}_i} \left(A[u, v] - \frac{D(u) \cdot D(v)}{2|\mathcal{E}|} \right). \tag{57}$$

Furthermore, the strength of the overall community detection result $\mathcal{C} = \{\mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_k\}$ can be defined as the *modularity* of the communities as follows.

DEFINITION 20. *(Modularity): Given the community detection result $\mathcal{C} = \{\mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_k\}$, the modularity of the community structure is defined as*

$$Q(\mathcal{C}) = \frac{1}{2|\mathcal{E}|} \sum_{\mathcal{U}_i \in \mathcal{C}} \sum_{u,v \in \mathcal{U}_i} \left(A[u,v] - \frac{D(u) \cdot D(v)}{2|\mathcal{E}|} \right). \tag{58}$$

The *modularity* concept effectively measures the strength of the detected community structure. Generally, for a community structure with a larger *modularity* score, it indicates a good community detection result.

Another way to explain the *modularity* is from the number of links within and across communities. By rewriting the above *modularity* equation, we can have

$$\begin{aligned}
& Q(\mathcal{C}) && (59)\\
&= \frac{1}{2|\mathcal{E}|} \sum_{\mathcal{U}_i \in \mathcal{C}} \sum_{u,v \in \mathcal{U}_i} \left(A[u,v] - \frac{D(u) \cdot D(v)}{2|\mathcal{E}|} \right) && (60)\\
&= \frac{1}{2|\mathcal{E}|} \left(\sum_{\mathcal{U}_i \in \mathcal{C}} \sum_{u,v \in \mathcal{U}_i} A[u,v] - \sum_{\mathcal{U}_i \in \mathcal{C}} \sum_{u,v \in \mathcal{U}_i} \frac{D(u) \cdot D(v)}{2|\mathcal{E}|} \right) && (61)\\
&= \frac{1}{2|\mathcal{E}|} \left(\sum_{\mathcal{U}_i \in \mathcal{C}} \sum_{u,v \in \mathcal{U}_i} A[u,v] - \frac{1}{2|\mathcal{E}|} \sum_{\mathcal{U}_i \in \mathcal{C}} \sum_{u \in \mathcal{U}_i} D(u) \sum_{u \in \mathcal{U}_i} D(v) \right) && (62)\\
&= \frac{1}{2|\mathcal{E}|} \left(\sum_{\mathcal{U}_i \in \mathcal{C}} \sum_{u,v \in \mathcal{U}_i} A[u,v] - \frac{1}{2|\mathcal{E}|} \sum_{\mathcal{U}_i \in \mathcal{C}} (\sum_{u \in \mathcal{U}_i} D(u))^2 \right). && (63)
\end{aligned}$$

In the above equation, term $\sum_{u,v \in \mathcal{U}_i} A[u,v]$ denotes the number of links connecting users within the community $\mathcal{U}_i$ (which will be 2 times the intra-community links for undirected networks, as each link will be counted twice). Term $\sum_{u \in \mathcal{U}_i} D(u)$ denotes the sum of node degrees in community $\mathcal{U}_i$, which equals to the number of intra-community and inter-community links connected to nodes in community $\mathcal{U}_i$. If there exist lots of inter-community links, then the *modularity* measure will have a smaller value. On the other hand, if the inter-community links are very rare, the *modularity* measure will have a larger value. Therefore, maximizing the community *modularity* measure is equivalent to minimizing the inter-community link numbers.

The *modularity* measure can also be represented with linear algebra equations. Let matrix $\mathbf{A}$ denote the adjacency matrix of the network, and vector $\mathbf{d} \in \mathbb{R}^{|\mathcal{V}| \times 1}$ denote the degrees of nodes in the network. The *modularity matrix* can be defined as

$$\mathbf{B} = \mathbf{A} - \frac{\mathbf{d}\mathbf{d}^\top}{2|\mathcal{E}|}. \tag{64}$$

Let matrix $\mathbf{H} \in \{0,1\}^{|\mathcal{V}| \times k}$ denotes the communities that users in $\mathcal{V}$ belong to. In real application, such a binary constraint can be relaxed to allow real value solutions for matrix $\mathbf{H}$. The optimal community detection result can be obtained by solving the following objective function

$$\max \frac{1}{2|\mathcal{E}|} \text{Tr}(\mathbf{H}^\top \mathbf{B} \mathbf{H}) \tag{65}$$

$$s.t.\ \mathbf{H}^\top \mathbf{H} = \mathbf{I}, \tag{66}$$

where constraint $\mathbf{H}^\top \mathbf{H} = \mathbf{I}$ ensures there are not overlap in the community detection result.

The above objective function looks very similar to the objective function of *spectral clustering* to be introduced in the next section. After obtaining the optimal $\mathbf{H}$, the communities can be obtained by applying the K-Means algorithm to $\mathbf{H}$ to determine the cluster labels of each node in the network.

6.1.3 *Spectral Clustering and Community Detection*

In the community detection process, besides maximizing the proximity of nodes belonging to the same communities (as introduced in Section *6.1.1*), minimizing the connections among nodes in different clusters is also an important factor. Different from the previous proximity based community detection algorithms, another way to address the community detection problem is from the cost perspective. Partition the nodes into different clusters will cut the links among the clusters. To ensure the nodes partitioned into different clusters have less connections with each other, the number of links to be cut in the community detection process should be as small as possible [59; 71].

Cut: Formally, given the community structure $\mathcal{C} = \{\mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_k\}$ detected from network G. The number of links cut [59] between communities $\mathcal{U}_i, \mathcal{U}_j \in \mathcal{C}$ can be represented as

$$cut(\mathcal{U}_i, \mathcal{U}_j) = \sum_{u \in \mathcal{U}_i} \sum_{v \in \mathcal{U}_j} I(u,v), \tag{67}$$

where function $I(u,v) = 1$ if $(u,v) \in \mathcal{E}$; otherwise, it will be 0. The total number of links cut in the partition process can be represented as

$$cut(\mathcal{C}) = \sum_{\mathcal{U}_i \in \mathcal{C}} cut(\mathcal{U}_i, \bar{\mathcal{U}_i}), \tag{68}$$

where set $\bar{\mathcal{U}_i} = \mathcal{C} \setminus \mathcal{U}_i$ denotes the communities except $\mathcal{U}_i$.

By minimizing the cut cost introduced in the partition process, the optimal community detection result can be obtained with the minimum number of cross-community links. However, as introduced in [59; 71], by minimizing the cut of edges across clusters, the results may involve high imbalanced communities, some community may involve one single node. Such a problem will be much more severe when it comes to the real-world social network data. In the following part of this section, we will introduce two other cost measures that can help achieve more balanced community detection results.

Ratio-Cut and Normalized-Cut: As shown in the example, the minimum cut cost treat all the links in the network equally, and can usually achieve very imbalanced partition results (e.g., a singleton node as a cluster) when applied in the real-world community detection problem. To overcome such a disadvantage, some models have been proposed to take the community size into consideration. The community size can be calculated by counting the number of nodes or links in each community, which will lead to two new cost measures: *ratio-cut* and *normalized-cut* [59; 71].

Formally, given the community detection result $\mathcal{C} = \{\mathcal{U}_1, \mathcal{U}_2, \cdots, \mathcal{U}_k\}$ in network G, the *ratio-cut* and *normalized-cut* costs introduced in the community detection result can be defined as follows respectively.

$$ratio - cut(\mathcal{C}) = \frac{1}{k} \sum_{\mathcal{U}_i \in \mathcal{C}} \frac{cut(\mathcal{U}_i, \bar{\mathcal{U}_i})}{|\mathcal{U}_i|}, \tag{69}$$

where $|\mathcal{U}_i|$ denotes the number of nodes in community $\mathcal{U}_i$.

$$ncut(\mathcal{C}) = \frac{1}{k} \sum_{\mathcal{U}_i \in \mathcal{C}} \frac{cut(\mathcal{U}_i, \bar{\mathcal{U}_i})}{vol(\mathcal{U}_i)}, \tag{70}$$

where $vol(\mathcal{U}_i)$ denotes the degree sum of nodes in community $\mathcal{U}_i$.

As shown in the above example, from the computed costs, we find that the community detected in plot C achieves much lower ratio-cut and ncut costs compared with those in plots B and D. Compared against the regular *cut* cost, both *ratio-cut* and *normalized-cut* prefer a balanced partition of the social network.

Spectral Clustering: Actually the objective function of both *ratio-cut* and *normalized-cut* can be unified as the following linear algebra equation

$$\min_{\mathbf{H} \in \{0,1\}^{|\mathcal{V}| \times k}} \mathrm{Tr}(\mathbf{H}^\top \bar{\mathbf{L}} \mathbf{H}), \tag{71}$$

where matrix $\mathbf{H} \in \{0,1\}^{|\mathcal{V}| \times k}$ denotes the communities that users in $\mathcal{V}$ belong to.

Let $\mathbf{A} \in \{0,1\}^{|\mathcal{V}| \times |\mathcal{V}|}$ denote the social adjacency matrix of the network, and the corresponding diagonal matrix of $\mathbf{A}$ can be represented as matrix $\mathbf{D}$, where $\mathbf{D}$ has value $D(i,i) = \sum_j A(i,j)$ on its diagonal. The Laplacian matrix of the network adjacency matrix $\mathbf{A}$ can be represented as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. Depending on the specific measures applied, matrix $\bar{\mathbf{L}}$ can be represented as

$$\bar{\mathbf{L}} = \begin{cases} \mathbf{L}, & \text{for ratio-cut measure,} \\ \mathbf{D}^{\frac{-1}{2}} \mathbf{L} \mathbf{D}^{\frac{-1}{2}}, & \text{for normalized-cut measure.} \end{cases} \tag{72}$$

The binary constraint on the variable $\mathbf{H}$ renders the problem a non-linear integer programming problem, which is very hard to solve. One common practice to learn the variable $\mathbf{H}$ is to apply spectral relaxation to replace the binary constraint with the orthogonality constraint.

$$\min \mathrm{Tr}(\mathbf{H}^\top \bar{\mathbf{L}} \mathbf{H}), \tag{73}$$

$$s.t. \mathbf{H}^\top \mathbf{H} = \mathbf{I}. \tag{74}$$

As proposed in [59], the optimal solution $\mathbf{H}^*$ to the above objective function equals to the eigen-vectors corresponding to the k smallest eigen-values of matrix $\bar{\mathbf{L}}$.

6.2 Mutual Community Detection

Besides the knowledge transfer from developed networks to the emerging networks to overcome the cold start problem [95], information in developed networks can also be transferred mutually to help refine the detected community structure detected from each of them. In this section, we will introduce the mutual community detection problem across multiple aligned heterogeneous networks and introduce a new cross-network mutual community detection model MCD. To refine the community structures, a new concept named *discrepancy* is introduced to help preserve the consensus of the community detection result of the shared anchor users according to [97].

For the given two aligned heterogeneous networks $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(1)}$, the *Mutual Community Detection* problem aims to obtain the optimal communities $\mathcal{C}^{(1)} = \{U_1^{(1)}, U_2^{(1)}, \cdots, U_{k^{(1)}}^{(1)}\}$ and $\mathcal{C}^{(2)} = \{U_1^{(2)}, U_2^{(2)}, \cdots, U_{k^{(2)}}^{(2)}\}$ of these two networks respectively. Users in each detected social community are more densely connected with each other than with users in other communities. Instead of the propagation based social intimacy score computation among users, MCD proposes to use the meta paths introduced in Section 3 to utilize both direct and indirect connections among users in closeness scores calculation. With full considerations of the network characteristics, MCD exploits the information in aligned networks to refine and disambiguate the community structures of the multiple networks concurrently based on a novel concept *community discrepency*. More detailed information about the MCD model will be introduced as follows.

6.2.1 Discrepancy

By maximizing the consensus (i.e., minimizing the "*discrepancy*") of the clustering results about the anchor users in multiple partially aligned networks, model MCD will be able to refine the clustering results of the anchor users with information in other aligned networks mutually. The confidence scores for each user belong to these communities can be represented as matrices $\mathbf{H}^{(1)}$ and $\mathbf{H}^{(2)}$. Matrix $\mathbf{H}^{(1)} = [\mathbf{h}_1^{(1)}, \mathbf{h}_2^{(1)}, \ldots, \mathbf{h}_n^{(1)}]^\top$, $n = |\mathcal{U}|$, $\mathbf{h}_i^{(1)} = (h_{i,1}, h_{i,2}, \ldots, h_{i,k})$ and $h_{i,j}$ denotes the confidence that $u_i^{(1)} \in \mathcal{U}^{(1)}$ is in cluster $U_j^{(1)} \in \mathcal{C}^{(1)}$. And it is similar for matrix $\mathbf{H}^{(2)}$.

Let u_i and u_j be two anchor users in the network, whose accounts in $G^{(1)}$ and $G^{(2)}$ are $u_i^{(1)}$, $u_i^{(2)}$, $u_j^{(1)}$ and $u_j^{(2)}$ respectively. If users $u_i^{(1)}$ and $u_j^{(1)}$ are partitioned into the same cluster in $G^{(1)}$ but their corresponding accounts $u_i^{(2)}$ and $u_j^{(2)}$ are partitioned into different clusters in $G^{(2)}$, then it will lead to a *discrepancy* [97; 57] between the clustering results of $u_i^{(1)}$, $u_i^{(2)}$, $u_j^{(1)}$ and $u_j^{(2)}$ in aligned networks $G^{(1)}$ and $G^{(2)}$.

The discrepancy between the clustering results of u_i and u_j across aligned networks $G^{(1)}$ and $G^{(2)}$ is defined as the difference of confidence scores of u_i and u_j being partitioned in the same cluster across aligned networks. Considering that in the clustering results, the confidence scores of $u_i^{(1)}$ and $u_j^{(1)}$ ($u_i^{(2)}$ and $u_j^{(2)}$) being partitioned into $k^{(1)}$ ($k^{(2)}$) clusters can be represented as vectors $\mathbf{h}_i^{(1)}$ and $\mathbf{h}_j^{(1)}$ ($\mathbf{h}_i^{(2)}$ and $\mathbf{h}_j^{(2)}$) respectively, while the confidences that u_i and u_j are in the same cluster in $G^{(1)}$ and $G^{(2)}$ can be denoted as $\mathbf{h}_i^{(1)}(\mathbf{h}_j^{(1)})^T$ and $\mathbf{h}_i^{(2)}(\mathbf{h}_j^{(2)})^T$. Formally, the discrepancy of the clustering results about u_i and u_j is defined to be $d_{ij}(\mathcal{C}^{(1)}, \mathcal{C}^{(2)}) = \left(\mathbf{h}_i^{(1)}(\mathbf{h}_j^{(1)})^T - \mathbf{h}_i^{(2)}(\mathbf{h}_j^{(2)})^T\right)^2$ if u_i, u_j are both anchor users; and $d_{ij}(\mathcal{C}^{(1)}, \mathcal{C}^{(2)}) = 0$ otherwise. Furthermore, the discrepancy of $\mathcal{C}^{(1)}$ and $\mathcal{C}^{(2)}$ will be:

$$d(\mathcal{C}^{(1)}, \mathcal{C}^{(2)}) = \sum_i^{n^{(1)}} \sum_j^{n^{(2)}} d_{ij}(\mathcal{C}^{(1)}, \mathcal{C}^{(2)}), \tag{75}$$

where $n^{(1)} = |\mathcal{U}^{(1)}|$ and $n^{(2)} = |\mathcal{U}^{(2)}|$. In the definition, non-anchor users are not involved in the discrepancy calculation.

6.2.2 Normalized Discrepancy

However, considering that $d(\mathcal{C}^{(1)}, \mathcal{C}^{(2)})$ is highly dependent on the number of anchor users and anchor links between $G^{(1)}$ and $G^{(2)}$, minimizing $d(\mathcal{C}^{(1)}, \mathcal{C}^{(2)})$ can favor highly consented clustering results when the anchor users are abundant but have no significant effects when the anchor users are very rare. To solve this problem, model MCD proposes to minimize the *normalized discrepancy* instead.

The normalized discrepancy measure computes the differences of clustering results in two aligned networks as a fraction of the discrepancy with regard to the number of anchor users across partially aligned networks:

$$nd(\mathcal{C}^{(1)}, \mathcal{C}^{(2)}) = \frac{d(\mathcal{C}^{(1)}, \mathcal{C}^{(2)})}{(|A^{(1,2)}|)\,(|A^{(1,2)}| - 1)}. \tag{76}$$

Optimal consensus clustering results of $G^{(1)}$ and $G^{(2)}$ will be:

$$\hat{\mathcal{C}}^{(1)}, \hat{\mathcal{C}}^{(2)} = \arg \min_{\mathcal{C}^{(1)}, \mathcal{C}^{(2)}} nd(\mathcal{C}^{(1)}, \mathcal{C}^{(2)}). \tag{77}$$

The normalized-discrepancy objective function can also be represented with the *clustering results confidence matrices* $\mathbf{H}^{(1)}$ and $\mathbf{H}^{(2)}$ as well. Meanwhile, considering that the networks studied in this section are partially aligned, matrices $\mathbf{H}^{(1)}$ and $\mathbf{H}^{(2)}$ contain the results of both anchor users and non-anchor users, while non-anchor users should not be involved in the discrepancy calculation according to the definition of discrepancy. The introduced model proposes to prune the results of the non-anchor users with the *anchor transition matrix* between the networks. By minimizing the *normalized discrepancy* together with the community detection cost terms of networks $G^{(1)}$ and $G^{(2)}$, MCD can learn the consensus community structures of multiple social networks mutually.

7. INFORMATION DIFFUSION

Social influence can be widely spread among people, and information exchange has become one of the most important social activities in the real world. The creation of the Internet and online social networks has rapidly facilitated the communication among people. Via the interactions among users in online social networks, information can be propagated from one user to other users. For instance, in recent years, online social networks have become the most important social occasion for news acquisition, and many outbreaking social events can get widely spread in the online social networks at a very fast speed. People as the multi-functional "sensors" can detect different kinds of signals happening in the real world, and write posts to report their discoveries to the rest of the world via the online social networks.

In this section, we will study the information diffusion process in the online social networks. *Diffusion* denotes the spreading process of certain entities (like information, idea, innovation, even heat in physics and disease in bio-medical science) through certain channels among the target object group in a system. The entities to be spread, the channels available, the target object group and the system can all affect the diffusion process and lead to different diffusion observations.

Depending on the system where the diffusion process is originally studied, the diffusion models can be divided into (1) information diffusion models in social networks [29; 103], (2) viral spreading in the bio-medical system [53; 12], and (3) heat diffusion in physical system [47; 8]. We will take the information diffusion in online social networks as one example. The channels for information diffusion belong to certain sources, like online world diffusion channels and offline world diffusion channels, or diffusion channels in different social networks. Meanwhile, depending on the diffusion channels and sources available, the diffusion models include (1) single-channel diffusion model [92; 29], (2) single source multi-channel diffusion model [82], (3) multi-source single-channel diffusion model [81; 77], and (4) multi-source multi-channel diffusion model [78; 103; 79]. Based on the categories of topics to be spread in the online social networks, the diffusion models can be categorized into (1) single topic diffusion [29; 78], (2) multiple intertwined topics concurrent diffusion [103; 92; 34; 13; 7].

In the following part of this section, we will introduce different kinds of diffusion models proposed to depict how information propagates among users in online social networks. We will first talk about the classic diffusion models proposed for the single-network single channel scenario, including the *threshold based models*, *cascades based models*, *heat diffusion based models* and *viral diffusion based models*. After that, the random walk based cross-network diffusion model will be introduced.

7.1 Traditional Information Diffusion Models

The "*diffusion*" phenomenon has been observed in different disciplines, like social science, physics, and bio-medical science. Various diffusion models have been proposed in these areas already. In this part, we will provide a brief introduction to these models, and introduce how to apply or adapt them for describe information diffusion process in online social networks.

Let $G = (\mathcal{V}, \mathcal{E})$ represent the network structure, based on which we want to study the information diffusion problem. Formally, given a user node $u \in \mathcal{V}$, we can represent the set of neighbors of u as $\Gamma(u)$. Each user node in the network G will have an indicator denoting whether the user has been activated or not. We will use notation $s(u) = 1$ to denote that user u has been activated, and $s(u) = 0$ to represent that u is still inactive. Initially, all the users are inactive to a certain information. Information can be propagated from an initial influence seed user set $\mathcal{S} \subset \mathcal{V}$ who are exposed to and activated by the information at the very beginning. At a timestamp in the diffusion process, given user u's neighbor, we can represent the subset of the active neighbors as $\Gamma^a(u) = \{v | v \in \Gamma(u), s(v) = 1\}$. The set of inactive neighbors can be represented as $\Gamma^i(u) = \Gamma(u) \setminus \Gamma^a(u)$. Generally, the information diffusion process will stop if no new activation is available.

7.1.1 Linear Threshold (LT) Models

In this subsection, we will introduce the threshold models, and will use *linear threshold model* as an example to illustrate such a kind of models. Several different variants of the *linear threshold models* will be briefly introduced here as well.

Generally, the *threshold models* assume that individuals have a unique threshold indicating the minimum amount of required information for them to be activated by certain information. Information can propagate among the users, and the information amount is determined by the closeness of the users. Close friends can influence each other much more than regular friends and strangers. If the information propagated from other users in the network surpass the threshold of a certain user, the user will turn to an activated status and also start to influence other users. Therefore, the threshold values can determine the performance of users in the online social networks. Depending on the setting of the thresholds as well as the amount of information propagated among the users, the *threshold models* have different variants.

LT Model: In the *linear threshold* (LT) model [29], each user has a unique threshold denoting the minimum required information to active the user. Formally, the threshold of user u can be represented as $\theta_u \in [0, 1]$. In the simulation experiments, the threshold values are normally selected from the uniform distribution $U(0, 1)$. Meanwhile, for each user pair, like $u, v \in \mathcal{V}$, information can be propagated between them. As mentioned before, close friends will have larger influence on each other compared with regular friends and strangers. Formally, the amount of information users u can send to v is denoted as weight $w_{u,v} \in [0, 1]$. Generally, the total amount of informations can send out is bounded. For instance, in the LT model, the total amount of information user u can send out is bounded by 1, i.e., $\sum_{v \in \Gamma(u)} w_{u,v} \leq 1$. Different ways have been proposed to define the specific value of the weight $w_{u,v}$ value, and in many of the cases $w_{u,v}$ can be different from $w_{v,u}$ since the information each user can send out can be different. However, in many other cases, to simplify the setting, for the same user pair, $w_{u,v}$ and $w_{v,u}$ are usually assigned with the same value. For instance, in some LT models, Jaccard's Coefficient is applied to calculate the closeness between the user pairs which will be used as the weight value.

In the LT model, the information sent from the neighbors to user u can be aggregated with linear summation. For instance, the total amount of information user u can receive from his/her neighbors can be denoted as $\sum_{v \in \Gamma(u)} w(v, u) s(v)$ or $\sum_{v \in \Gamma^a u} w(v, u)$. To check whether a user can be activated or not, LT model will only need to check whether the following equation holds or not,

$$\sum_{v \in \Gamma^a u} w(v, u) \geq \theta_u. \tag{78}$$

It denotes whether the received information surpasses the activation threshold of user u or not. Here, we also need to notice that inactive neighbors will not send out information, and only the active neighbors can send out information. The information provided so far shows the critical details of the LT model. Next, we will show the general framework of the LT model to illustrate how it works.

In the LT model, the initial activated seed user set can be represented as $\mathcal{S}$, users in which can start the propagation of information to their neighbors. Generally, information propagates within the network step by step.

- *Diffusion Starts*: At step 0, only the seed users in $\mathcal{S}$ are active, and all the remaining users have inactive status.

- *Diffusion Spreads*: At step $t(t > 0)$, for each user u, if the information propagated from u's active neighbors is greater than the threshold of u, i.e., $\sum_{v \in \Gamma^a u} w(v,u) \geq \theta_u$, u will be activated with status $s(u) = 1$. All activated users will remain active in the coming rounds, and can send information to their neighbors. Active user cannot be activated again.

- *Diffusion Ends*: If no new activation happens in step t, the diffusion process will stop.

Other Threshold Models: The LT model assumes the cumulative effects of information propagated from the neighbors, and can illustrate the basic information diffusion process among users in the online social networks. The LT model has been well analyzed, and many other variant models have been proposed as well. Depending on the assignment of the threshold and weight values, many other different diffusion models can all be reduced to a special case of the LT model.

- **Majority Threshold Model**: Different from the LT mode, in *majority threshold model* [10], an inactive user u can be activated if majority of his/her neighbors are activated. The *majority threshold model* can be reduced to the LT model in the case that: (1) the influence weight between any friends (u, v) in the network is assigned with value 1; (2) the threshold of any user u is set as $\frac{1}{2}D(u)$, where $D(u)$ denotes the degree of node u in the network. For the nodes with large degrees, like the central node in the star-structured diagram, their activation will lead to the activation of lots of surrounding nodes in the network.

- **k-Threshold Model**: Another diffusion model similar to the LT model is called the *k-threshold diffusion model* [10], in which users can be activated of at least k of his/her neighbors are active. The *k-threshold model* is equivalent to the LT model with settings (1) the influence weight between any friend pairs (u, v) in the network is assigned with value 1; and (2) the activation thresholds of all the users are assigned with a shared value k. For each user u, if k of his/her neighbors have been activated, u will be activated.

  Depending on the values of k, the *k-threshold model* will have different performance. When $k = 1$, a user will be activated of at least one of his/her neighbor is active. In such a case, all the users in the same connected components with the initial seed users will be activated finally. When k is a very large value and even greater than the large node degree, e.g., $k > \max_{u \in \mathcal{V}} D(u)$, no nodes can be activated. When k is a medium value, some of the users will be activated as the information propagates, but the other users with less than k neighbors will never be activated.

7.1.2 Independent Cascade (IC) Model

An information cascade occurs when a people observe the actions of others and then engage in the same acts. Cascade clearly illustrates the information propagation routes, and the activating actions performed for users to their neighbors. In this part, we will talk about the cascade based models and use the *independent cascade* (IC) model as an example to illustrate the model architecture.

IC Model: In the diffusion process, about one certain target user, multiple activation trials can be performed by his/her neighbors. In the *independent cascade* model [29], each activation is performed independently regardless of the historical unsuccessful trials. The activation trials are performed step by step. When user u who has been activated in the previous step and tries to activate user v in the current step, the success probability is denoted as $p_{u,v} \in [0, 1]$. Generally, if users u and v are close friends, the activation probability will be larger compared with regular friends and strangers. The specific activation probability values is usually correlated with the social closeness between users u and v, which can also be defined based the Jaccard's Coefficient in the simulation. The activation trials will only happen among the users who are friends. If u succeeds in activating v, then user v will change his/her status to "*active*" and will remain in the status in the following steps. However, if u fails to activate v, u will lose the chance and cannot perform the activation trials any more.

In IC model, the activation trials are performed by flipping a coin with certain probabilities, whose result is uncertain. Even with the same provided initial seed user set $\mathcal{S}$, the number of users who will be activated by the seed users can be different if we running the IC model twice. Formally, we can represent the set of activated users by the seed users as $\mathcal{V}^a \subset \mathcal{V}$. Therefore, in the experimental simulations, we usually run the diffusion model multiple times and calculate the average number of activated users, i.e., $|\mathcal{V}^a|$, to denote the expected influence achieved by the seed user set $\mathcal{S}$.

Other Cascade Models: Generally, the independent activation assumption renders the IC model the simplest cascade based diffusion models. In the real world, the diffusion process will be more complicated. For the users, who have been failed to be activated by many other users, it probably indicates that the user is not interested in the information. Viewed in such a perspective, the probability for the user to be activated will decrease as more activation trials have been performed. In this part, we will introduce another cascade based diffusion model, *decreasing cascade model* (DC) [30].

To illustrate the DC model more clearly and show it difference compared with the IC model, we use notation $P(u \to v|\mathcal{T})$ to represent the probability for user u to activate v given a set of users $\mathcal{T}$ have performed and failed the activation trials to v already. Let $\mathcal{T}, \mathcal{T}'$ denote two historical activation trial user set, where $\mathcal{T} \subseteq \mathcal{T}'$. In the IC model, we have

$$P(u \to v|\mathcal{T}) = P(u \to v|\mathcal{T}'). \tag{79}$$

In other words, every activation trial is independent with each other, and the activation probability will not be changed as more activation trials have been performed.

As introduced at the beginning of this subsection, the fact that users in set $\mathcal{T}$ fail to activate v indicates that v probably is not interested in the information, and the change for v to be activated afterwards will be lower. Furthermore, as more activation trials, e.g., users in $\mathcal{T}'$ are performed, the probability for u to active v will be decreased, i.e.,

$$P(u \to v|\mathcal{T}) \geq P(u \to v|\mathcal{T}'). \tag{80}$$

Intuitively, this restriction states that a contagious node's probability of activating some v decreases if more nodes have already attempted to activate v, and v is hence more "marketing-saturated". The DC model incorporates the IC model as a special case, and is more general in information diffusion modeling than the IC model.

7.1.3 Epidemic Diffusion Model

The threshold and cascade based diffusion models introduced in the previous part mostly assume that "once a user is activated, he/she will remain the active status forever". However, in the real world, these activated users can change their minds and the activated users can still have the chance to recover to the original status. In the bio-medical science, diffusion models have been studied for many years to model the spread of disease, and several *epidemic diffusion models* have been introduced already. In the disease propagation, people who are susceptible to the disease can be get infected by other people. After some time, many of these infected people can get recovered and become immune to the disease, while many other users can get recovered and get susceptible to the disease again. Depending on the people's reactions to the disease after recovery, several different *epidemic diffusion models* [51] have been proposed already.

Susceptible-Infected-Recovered (SIR) Diffusion Model: The SIR model was proposed by W. O. Kermack and A. G. McKendrick in 1927 to model the infectious diseases, which consider a fixed population with three main categories: *susceptible* (S), *infected* (I), and *recovered* (R). As the disease propagates, the individual status can change among {S, I, R} following flow:

$$S \rightarrow I \rightarrow R. \tag{81}$$

In other words, the individuals who are susceptible to the disease can get infected, while those infected individuals also have the chance to recover from the disease as well.

In this part, we will use the SIR model to describe the information cascading process in online social networks. Let $\mathcal{V}$ denote the set of users in the network. We introduce the following notations to represent the number of users in different categories:

- $S(t)$: the number of users who are *susceptible* to the information at time t, but have not gotten *infected* yet.
- $I(t)$: the number of users who are currently *infected* by the information, and can spread the information to others in the *susceptible* catetory.
- $R(t)$: the number of users who have been infected and already recovered from the information infection. The users are immune to the information will not be infected again.

Based on the above notations, we have the following equations hold in the SIR model.

$$S(t) + I(t) + R(t) = |\mathcal{V}|, \tag{82}$$

$$\frac{\mathrm{d}S(t)}{\mathrm{d}t} + \frac{\mathrm{d}I(t)}{\mathrm{d}t} + \frac{\mathrm{d}R(t)}{\mathrm{d}t} = 0, \tag{83}$$

where,

$$\begin{cases} \frac{\mathrm{d}S(t)}{\mathrm{d}t} = -\beta S(t) I(t), \\ \frac{\mathrm{d}I(t)}{\mathrm{d}t} = \beta S(t) I(t) - \gamma I(t), \\ \frac{\mathrm{d}R(t)}{\mathrm{d}t} = \gamma I(t). \end{cases} \tag{84}$$

Other Epidemic Diffusion Model

In some cases, the users cannot get immune to the information and don't exist the *recovery* status actually. For the users, who get infected, they can go to the *susceptible* status and can get *infected* again in the future. To model such a phenomenon, another diffusion model very similar to the SIR model has been proposed, which is called the Susceptible-Infected-Susceptible (SIS) model. Furthermore, in another variant of SIR, the individuals in the *recovery* category can lose the immunity and transit to the *susceptible* category and have the potential to be infected again. Therefore, the individual status flow will be

$$S \rightarrow I \rightarrow R \rightarrow S. \tag{85}$$

And the model is also named as the SIRS diffusion model.

Besides these epidemic diffusion models introduced in this subsection, there also exist many different version of the epidemic diffusion models, which considers many other factors in the diffusion process, like the birth/death of individuals. It is also very common in the real-world online social networks, since new users will join in the social network, and existing users will also delete their account and get removed from the social network. Involving such factors will make the diffusion model more complex, and we will not introduce them here due to the limited space. More information about these epidemic diffusion models is available in [49; 51].

7.1.4 Heat Diffusion Models

Heat diffusion is a well observed physical phenomenon. Generally, in a medium, heat will always diffuses from regions with a high temperature to the region with a lower temperature. Recently, many works have applied the heat diffusion to model the information propagation in online social networks. In this subsection, we will talk about the *heat diffusion model* and introduce how to adapt it to model the information diffusion in online social networks.

General Heat Diffusion: Throughout a geometric manifold, let function $f(x, t)$ denote the temperature at location x at time t, and we can represent the initial temperature at different locations as $f_0(x)$. The heat flows with initial conditions can be described by the following second order differential equation

$$\begin{cases} \frac{\partial f(x,t)}{\partial t} - \Delta f(x,t) = 0 \\ f(x, 0) = f_0(x), \end{cases} \tag{86}$$

where $\Delta f(x, t)$ is a *Laplace-Beltrami operator* on function $f(x, t)$. Many existing works on the heat diffusion studies are mainly focused on the heat kernel matrix. Formally, let $\mathbf{K}_t$ denote the heat kernel matrix at timestamp t, which describes the heat diffusion among different regions in the medium. In the matrix, entry $K_t(x, y)$ denotes the heat diffused from the original position y to position x at time t. However, it is very difficult to represent the medium as a regular geometry with a known dimension. In the next part, we will introduce how to apply the heat diffusion observations to model the information diffusion in the network-structured graph data.

Heat Diffusion Model: Given a homogeneous network $G = (\mathcal{V}, \mathcal{E})$, for each node $u \in \mathcal{V}$ in the network, we can represent the information at u in timestamp t as $f(u, t)$. The initial information available at each of the node can be denoted as $f(u, 0)$. The information can be propagated among the nodes in the network if there exists a pipe (i.e., a link) between them. For instance, with a link $(u, v) \in \mathcal{E}$ in the network, information can be propagated between u and v.

Generally, in the diffusion process, the amount of information propagated between different nodes in the network depends on (1) the difference of information available at them, and (2) the thermal conductivity-the heat diffusion coefficient α. For instance, at timestamp t, we can represent the amount of information reaching nodes $u, v \in \mathcal{V}$ as $f(u, t)$ and $f(v, t)$. If $f(u, t) > f(v, t)$, information tends to propagate from u to v in the network, and the amount of information propagated is $\alpha \cdot (f(u, t) - f(v, t))$, and the propagation direction will be reversed if $f(u, t) < f(v, t)$. The information amount changes at node u at timestamps t and $t + \Delta t$ can be represented as

$$\frac{f(u,t+\Delta t)-f(u,t)}{\Delta t} = -\sum_{v\in\Gamma(u)} \alpha\cdot(f(u,t)-f(v,t)) \,. \quad (87)$$

Let's use vector $\mathbf{f}(t)$ to represent the amount of information available at all the nodes in the network at timestamp t. The above information amount changes can be rewritten as

$$\frac{\mathbf{f}(t+\Delta t)-\mathbf{f}(t)}{\Delta t} = \alpha \mathbf{H}\mathbf{f}(t), \quad (88)$$

where in the matrix $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, entry $H(u,v)$ has value

$$H(u,v) = \begin{cases} 1, & \text{if } (u,v)\in\mathcal{E} \vee (v,u)\in\mathcal{E}, \\ -D(u), & \text{if } u=v, \\ 0, & \text{otherwise,} \end{cases} \quad (89)$$

where $D(u)$ denotes the degree of node u in the network.
In the limit case $\Delta t \to 0$, we can rewrite the equation as

$$\frac{\mathrm{d}\mathbf{f}(t)}{\mathrm{d}t} = \alpha\mathbf{H}\mathbf{f}(t). \quad (90)$$

Solving the function, we can represent the amount of information at each node in the network as

$$\mathbf{f}(t) = \exp^{t\alpha\mathbf{H}}\mathbf{f}(0) \quad (91)$$

$$= \left(\mathbf{I} + \alpha t\mathbf{H} + \frac{\alpha^2t^2}{2!}\mathbf{H}^2 + \frac{\alpha^3t^3}{3!}\mathbf{H}^3 + \cdots\right)\mathbf{f}(0), \quad (92)$$

where term $\exp^{t\alpha\mathbf{H}}$ is called the diffusion kernel matrix, which can be expanded according to Taylor's theorem.

7.2 Random Walk based Diffusion Model

Different online social networks usually have their own characteristics, and users tend to have different status regarding the same information. For instance, information about personal entertainments (like movies, pop stars) can be widely spread among users in Facebook, and users interested in them will be activated very easily and also share the information to their friends. However, such a kind of information is relatively rare in the professional social network LinkedIn, where people seldom share personal entertainment to their colleagues, even though they may have been activated already in Facebook. What's more, the structures of these online social networks are usually heterogeneous, containing many different kinds of connections. Besides the direct follow relationships among the users, these diverse connections available among the users may create different types of communication channels for information diffusion. To model such an observation in information diffusion across multiple heterogeneous online social sites, in this part, we will introduce a new information diffusion model, IPATH [80], based on random walk.

7.2.1 *Intra-Network Propagation*

In a heterogeneous network, multi-typed and interconnected entities, such as images, videos and locations, can create various information propagation relations among users. We can represent the information diffusion routes among users via other information entities, which can be formally represented as the diffusion route set $\mathcal{R} = \{r_1, r_2, \ldots, r_m\}$, where m is the route number. Let's take the source network $G^{(s)} = (\mathcal{V}^{(s)}, \mathcal{E}^{(s)})$ as an example. For any diffusion route $r_i \in \mathcal{R}$, the adjacency matrix of r_i will be $\mathbf{A}_i^{(s)} \in \mathbb{R}^{|\mathcal{V}^{(s)}|\times|\mathcal{V}^{(s)}|}$, where $A_i^{(s)}(u,v)$ is a binary-value variable and $A_i^{(s)}(u,v) = 1$ iff u and v are connected with each other via relation r_i. The weighted diffusion matrix can be represented as

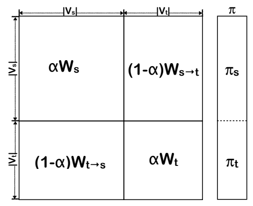

Figure 3: The weight matrix and the information distribution vector

the normalization of $\mathbf{W}_i^{(s)} = \mathbf{A}_i^{(s)}\mathbf{D}^{-1}$, where $\mathbf{D}^{-1}$ is a diagonal matrix with $D(u,u) = \sum_v^{|\mathcal{V}^{(s)}|} A_i^{(s)}(v,u)$, denoting the in-degree of u. In a similar way, we can represent the weighted diffusion matrices for other relations, which altogether can be represented as $\{\mathbf{W}_1^{(s)}, \mathbf{W}_2^{(s)}, \ldots, \mathbf{W}_m^{(s)}\}$. To fuse the information diffused from different relations, IPATH will linearly combine these weighted matrices as follows:

$$\mathbf{W}_s = \lambda_1 \times \mathbf{W}_1^{(s)} + \lambda_2 \times \mathbf{W}_2^{(s)} + \cdots + \lambda_m \times \mathbf{W}_m^{(s)}, \quad (93)$$

where λ_i denotes the aggregation weight of matrix corresponding to relation r_i. In a similar way, we can define the weight matrix $\mathbf{W}^{(t)}$ of the target network $G^{(t)}$.

7.2.2 *Inter-Network Propagation*

Across the aligned networks, information can propagate not only within networks but also across networks. Based on the known anchor links between networks $G^{(t)}$ and $G^{(s)}$, i.e., set $\mathcal{A}^{(s,t)}$, we can define the binary adjacency matrix $\mathbf{A}^{(s\to t)} \in \mathbb{R}^{|\mathcal{V}^{(s)}|\times|\mathcal{V}^{(t)}|}$, where $A^{(s\to t)}(u,v) = 1$ if $(u^{(s)}, v^{(t)}) \in \mathcal{A}^{(s,t)}$. IPATH assumes that each anchor user in $G^{(s)}$ only has one corresponding account in $G^{(t)}$. Therefore $\mathbf{A}^{(s\to t)}$ has been normalized and the weight matrix $\mathbf{W}^{(s\to t)} = \mathbf{A}^{(s\to t)}$, denoting the chance of information propagating from $G^{(s)}$ to $G^{(t)}$. Furthermore, we can represent the weighted diffusion matrix from networks $G^{(t)}$ to $G^{(s)}$ as $\mathbf{W}^{(t\to s)} = (\mathbf{W}^{(s\to t)})^\top$, considering that the anchor links are undirected.

Both the intra-network propagation relations, represented by weight matrices $\mathbf{W}^{(s)}$ and $\mathbf{W}^{(t)}$ in networks $G^{(s)}$ and $G^{(t)}$ respectively, and the inter-network propagation relations, represented by weight matrix $\mathbf{W}^{(s\to t)}$ and $\mathbf{W}^{(t\to s)}$, have been constructed already in the previous subsection. As shown in Figure 3, to model the cross-network information diffusion process involving both the intra- and inter-network relations simultaneously, IPATH proposes to combine these weighted diffusion matrices to build an integrated matrix $\mathbf{W} \in \mathbb{R}^{(|\mathcal{V}^{(s)}|+|\mathcal{V}^{(t)}|)^2}$. In the integrated matrix $\mathbf{W}$, the parameter $\alpha \in [0,1]$ denotes the probability that the message stay in the original network, thus $1-\alpha$ represents the chance of being transmitted across networks (i.e., the probability of activated anchor user passing the influence to the target network).

7.2.3 *The IPATH Information Propagation Model*

Let vector $\pi_k \in \mathbb{R}^{(|\mathcal{V}^{(s)}|+|\mathcal{V}^{(t)}|)}$ represent the information that users in $G^{(s)}$ and $G^{(t)}$ can receive after k steps. As shown in Figure 3, vector π_k consists of two parts $\pi_k = [\pi_k^{(s)}, \pi_k^{(t)}]$, where $\pi_k^{(s)} \in \mathbb{R}^{|\mathcal{V}^{(s)}|}$ and $\pi_k^{(t)} \in \mathbb{R}^{|\mathcal{V}^{(t)}|}$. The initial state of the vector can be denoted as π_0, which is defined based on the seed user set $\mathcal{Z}$ with function $g(\cdot)$ as follows:

$$\pi_0 = g(\mathcal{Z}), \text{where } \pi_0[u] \quad = \begin{cases} 1 & \text{if } u \in \mathcal{Z}, \\ 0 & \text{otherwise.} \end{cases} \quad (94)$$

Seed set $\mathcal{Z}$ can also be represented as $\mathcal{Z} = g^{-1}(\pi_0)$. Users from $G^{(s)}$ and $G^{(t)}$ both have the chance of being selected as seeds, but when the structure information of $G^{(t)}$ is hard to obtain, the seed users will be only chosen from $G^{(s)}$. In IPATH, the information diffusion process is modeled by *random walk*, because it is widely used in which the total probability of the diffusing through different relations remains constant 1 [68; 20]. Therefore, in the information propagation process, vector π will be updated stepwise with the following equation:

$$\pi^{(k+1)} = (1-\alpha) \times \mathbf{W}\pi_k + \alpha \times \pi_0, \tag{95}$$

where constant α denotes the probability of returning to the initial state. By keeping updating π according to (95) until convergence, we can present the stationary state of vector π to be π^*,

$$\pi^* = \alpha[\mathbf{I} - (1-\alpha)\mathbf{W}]^{-1}\pi_0, \tag{96}$$

where matrix $\mathbf{I} \in \{0,1\}^{(|\mathcal{V}^{(s)}|+|\mathcal{V}^{(t)}|)\times(|\mathcal{V}^{(s)}|+|\mathcal{V}^{(t)}|)}$ is an identity matrix. The value of entry $\pi^*[u]$ denotes the activation probability of u, and user u will be activated if $\pi^*[u] \geq \theta$, where θ denotes the threshold of accepting the message. In IPATH, parameter θ is randomly sampled from range $[0, \theta_{bound}]$. The threshold bound θ_{bound} is a small constant value, as the amount of information each user can get at the stationary state in IPATH can be very small (which is set as 0.01 in the experiments). In addition, we can further represent the activation status of user u as vector π', where

$$\pi'[u] = \begin{cases} 1 & \text{if } \pi^*[u] \geq \theta, \\ 0 & \text{otherwise.} \end{cases} \tag{97}$$

8. NETWORK EMBEDDING

In the concrete applications, great challenges exist in handling the network structured data with traditional machine learning algorithms, which usually take feature vector representation data as the input. A general representation of heterogeneous networks as feature vectors is desired for knowledge discovery from such complex network structured data. In recent years, many research works propose to embed the online social network data into a lower-dimensional feature space, in which the user node is represented as a unique feature vector, and the network structure can be reconstructed from these feature vectors. With the embedded feature vectors, classic machine learning models can be applied to deal with the social network data directly, and the storage space can be saved greatly.

In this section, we will talk about the *network embedding* problem, aiming at projecting the nodes and links in the network data in low-dimensional feature spaces. Depending on the application setting, existing graph embedding works can be categorized into the embedding of *homogeneous networks*, *heterogeneous networks*, and *multiple aligned heterogeneous networks*. Meanwhile, depending on the models being applied, current embedding works can be divided into the *matrix factorization based embedding*, *translation based embedding*, and *deep learning based embedding*.

In the following parts in this section, we will first introduce the *translation based graph embedding* models in Section 8.1, which are mainly proposed for the multi-relational knowledge graphs, including TransE [9], TransH [73] and TransR [40]. After that, in Section 8.2, we will introduce three homogeneous network embedding models, including DeepWalk [52], LINE [67] and node2vec [22]. Finally, we will talk about the model proposed for the multiple aligned heterogeneous network [93] in Section 8.3, where the anchor links are utilized to transfer information across different sites for mutual refinement of the embedding results mutually.

8.1 Translation based Network Embedding

Multi-relational data refers to the directed graphs whose nodes correspond to entities and links denote the relationships. The multi-relational data can be represented as a graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the node set and $\mathcal{E}$ represents the link set. For the link in the graph, e.g., $r = (h,t) \in \mathcal{E}$, the corresponding entity-relation can be represented as a triple (h,r,t), where h denotes the link initiator entity, t denotes the link recipient entity and r represents the link. The embedding problem studied in this section is to learn a feature representation of both entities and relations in the triples, i.e., h, r and t.

8.1.1 TransE

The TransE [9] model is an energy-based model for learning low-dimensional embeddings of entities and relations, where the relations are represented as the *translations* of entities in the embedding space. Given a entity-relation triple (h,r,t), the embedding feature representation of the entities and relations can be represented as vectors $\mathbf{h} \in \mathbb{R}^k$, $\mathbf{r} \in \mathbb{R}^k$ and $\mathbf{t} \in \mathbb{R}^k$ (k denotes the objective vector dimension). If the triple (h,r,t) holds, i.e., there exists a link r starting from h to t in the network, the corresponding embedding vectors $\mathbf{h} + \mathbf{r}$ should be as close to vector $\mathbf{t}$ as possible.

Let $\mathcal{S}^+ = \{(h,r,t)\}_{r=(h,t)\in\mathcal{E}}$ represents the set of positive training data, which contains the triples existing in the networks. The TransE model aims at learning the embedding features vectors of the entities h, t and the relation r, i.e., $\mathbf{h}$, $\mathbf{r}$ and $\mathbf{t}$. For the triples in the positive training set, we want to ensure the learnt embedding vectors $\mathbf{h}+\mathbf{r}$ is very close to $\mathbf{t}$. Let $d(\mathbf{h}+\mathbf{r}, \mathbf{t})$ denotes the distance between vectors $\mathbf{h}+\mathbf{r}$ and $\mathbf{t}$. The loss introduced for the triples in the positive training set can be represented as

$$\mathcal{L}(\mathcal{S}^+) = \sum_{(h,r,t)\in\mathcal{S}^+} d(\mathbf{h}+\mathbf{r}, \mathbf{t}). \tag{98}$$

Here the distance function can be defined in different ways, like the L_2 norm of the difference between vectors $\mathbf{h}+\mathbf{r}$ and $\mathbf{t}$, i.e.,

$$d(\mathbf{h}+\mathbf{r}, \mathbf{t}) = \left\|\mathbf{h}+\mathbf{r}-\mathbf{t}\right\|_2. \tag{99}$$

By minimizing the above loss function, the optimal feature representations of the entities and relations can be learnt. To avoid trivial solutions, like $\mathbf{0}$s for $\mathbf{h}$, $\mathbf{r}$ and $\mathbf{t}$, additional constraints that the L_2-norm of the embedding vectors of the entities should be 1 will be added in the function. Furthermore, a negative training set is also sampled to differentiate the learnt embedding vectors. For a triple $(h,r,t) \in \mathcal{S}^+$, the corresponding sampled negative training set can be denoted as $\mathcal{S}^-_{(h,r,t)}$, which contains the triples formed by replacing the initiator entity h or the recipient entity t with random entities. In other words, the negative training set $\mathcal{S}^-_{(h,r,t)}$ can be represented as

$$\mathcal{S}^-_{(h,r,t)} = \{(h',r,t)|h' \in \mathcal{V}\} \cup \{(h,r,t')|t' \in \mathcal{V}\}. \tag{100}$$

The loss function involving both the positive and negative training set can be represented as

$$\mathcal{L}(\mathcal{S}^+, \mathcal{S}^-) = \tag{101}$$

$$\sum_{(h,r,t)\in\mathcal{S}^+} \sum_{(h',r,t')\in\mathcal{S}^-_{(h,r,t)}} \max\left(\gamma + d(\mathbf{h}+\mathbf{r},\mathbf{t}) - d(\mathbf{h}'+\mathbf{r},\mathbf{t}'), 0\right), \tag{102}$$

where γ is a margin hyperparameter and $\max(\cdot, 0)$ will count the positive loss only.

The optimization is carried out by stochastic gradient descent (in minibatch mode). The embedding vectors of entities and relationships are initialized with a random procedure. At each iteration of the algorithm, the embedding vectors of the entities are normalized and a small set of triplets is sampled from the training set, which will serve as the training triplets of the minibatch. The parameters are then updated by taking a gradient step.

8.1.2 *TransH*

TransE is a promising method proposed recently, which is very efficient while achieving state-of-the-art predictive performance. However, in the embedding process, TransE fail to consider the *cardinality constraint* on the relations, like *one-to-one*, *one-to-many* and *many-to-many*. The TransH model [73] to be introduced in this part considers such properties on relations in the embedding process. Different from the other complex models, which can handle these properties but sacrifice efficiency, TransH achieves comparable time complexity as TransE. TransH models the relation as a hyperplane together with a translation operation on it, where the correlation among the entities can be effectively preserved.

In TransH, different from the embedding space of entities, the relations, e.g., r, is denoted as a transition vector $\mathbf{d}_r$ in the hyperplane $\mathbf{w}_r$ (a normal vector). For each of the triple (h, r, t), the embedding vector $\mathbf{h}$, $\mathbf{t}$ are fist projected to the hyperplane $\mathbf{w}_r$, whose corresponding projected vectors can be represented as $\mathbf{h}_\perp$ and $\mathbf{t}_\perp$ respectively. The vectors $\mathbf{h}_\perp$ and $\mathbf{t}_\perp$ can be connected by the translation vector $\mathbf{d}_r$ on the hyperplane. Depending on whether the triple appears in the positive or negative training set, the distance $d(\mathbf{h}_\perp + \mathbf{d}_r, \mathbf{t}_\perp)$ should be either minimized or maximized.

Formally, given the hyperplane $\mathbf{w}_r$, the projection vectors $\mathbf{h}_\perp$ and $\mathbf{t}_\perp$ can be represented as

$$\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h} \mathbf{w}_r, \tag{103}$$

$$\mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t} \mathbf{w}_r. \tag{104}$$

Furthermore, the L_2 norm based distance function can be represented as

$$d(\mathbf{h}_\perp + \mathbf{d}_r, \mathbf{t}_\perp) = \|(\mathbf{h} - \mathbf{w}_r \mathbf{h} \mathbf{w}_r) + \mathbf{d}_r - (\mathbf{t} - \mathbf{w}_r \mathbf{t} \mathbf{w}_r)\|_2^2 . \tag{105}$$

The variables to be learnt in the TransH model include the embedding vectors of all the entities, the hyperplane and translation vectors for each of the relations. To learn these variables simultaneously, the objective function of TransH can be represented as

$$\mathcal{L}(\mathcal{S}^+, \mathcal{S}^-) = \tag{106}$$

$$\sum_{(h,r,t) \in \mathcal{S}^+} \sum_{(h',r',t') \in \mathcal{S}^-_{(h,r,t)}} \max\left(\gamma + d(\mathbf{h}_\perp + \mathbf{d}_r, \mathbf{t}_\perp) - d(\mathbf{h}'_\perp + \mathbf{d}'_r, \mathbf{t}'_\perp), 0\right), \tag{107}$$

where $\mathcal{S}^-_{(h,r,t)}$ denotes the negative set constructed for triple (h, r, t). Different from TransE, TransH applies a different to sample the negative training triples with considerations of the relation *cardinality constraint*. For the relations with *one-to-many*, TransH will give more chance to replace the initiator node; and for the *many-to-one* relations, TransH will give more chance to replace the recipient node instead.

Besides the loss function, the variables to be learnt are subject to some constraints, like the embedding vector for entities is a normal vector; $\mathbf{w}_r$ and $\mathbf{d}_r$ should be orthogonal, and $\mathbf{w}_r$ is also a normal vector. We summarize the constraints of the TransH model as follows

$$\|\mathbf{h}\|_2 \leq 1, \|\mathbf{t}\|_2 \leq 1, \forall h, t \in \mathcal{V}, \tag{108}$$

$$\frac{|\mathbf{w}_r^\top \mathbf{d}_r|}{\|\mathbf{d}_r\|_2} \leq \epsilon, \forall r \in \mathcal{E}, \tag{109}$$

$$\|\mathbf{w}_r\|_2 \leq 1, \forall r \in \mathcal{E}. \tag{110}$$

The constraints can be relaxed as some penalty terms, which can be added to the objective function with a relatively large weight. The final objective function can be learnt with the stochastic gradient descent, and by minimizing the loss function, the model variables can be learned and we will get the final embedding results.

8.1.3 *TransR*

Both TransE and TransH introduced in the previous subsections assume embeddings of entities and relations within the same space $\mathbb{R}^k$. However, entities and relations are actually totally different objects, and they may be not capable to be represented in a common semantic space. To address such a problem, TransR [40] is proposed, which models the entities and relations in distinct spaces, i.e., the entity space and relation space, and performs the translation in relation space.

In TransR, given a triple (h, r, t), the entities h and t are embedded as vectors $\mathbf{h}, \mathbf{t} \in \mathbb{R}^{k_e}$, and the relation r is embedded as vector $\mathbf{r} \in \mathbb{R}^{k_r}$, where the dimension of the entity space and relation space are not the same, i.e., $k_e \neq k_r$. To project the entities from the entity space to the relation space, a projection matrix $\mathbf{M}_r \in \mathbb{R}^{k_e \times k_r}$ is defined in TransR. With the projection matrix, the projected entity embedding vectors can be defined as

$$\mathbf{h}_r = \mathbf{h}\mathbf{M}_r, \tag{111}$$

$$\mathbf{t}_r = \mathbf{t}\mathbf{M}_r. \tag{112}$$

The loss function is defined as

$$d(\mathbf{h}_r + \mathbf{r}, \mathbf{t}_r) = \|\mathbf{h}_r + \mathbf{r} - \mathbf{t}_r\|_2^2 . \tag{113}$$

The constraints involved in TransR include

$$\|\mathbf{h}\|_2 = 1, \|\mathbf{t}\|_2 = 1, \forall h, t \in \mathcal{V}, \tag{114}$$

$$\|\mathbf{h}\mathbf{M}_r\|_2 = 1, \|\mathbf{t}\mathbf{M}_r\|_2 = 1, \forall h, t \in \mathcal{V}, \tag{115}$$

$$\|\mathbf{w}_r\|_2 \leq 1, \forall r \in \mathcal{E}. \tag{116}$$

The negative training set $\mathcal{S}^-$ in TransR can be obtained in a similar way as TransH, where the variables can be learnt with the stochastic gradient descent. We will not introduce the information here to avoid content duplication.

8.2 Homogeneous Network Embedding

Besides the translation based network embedding models, in this section, we will introduce three embedding models for network data, including DeepWalk, LINE and node2vec. Formally, the networks studied in this part are all homogeneous networks, which is represented as $G = (\mathcal{V}, \mathcal{E})$. Set $\mathcal{V}$ denotes the set of nodes in the homogeneous network, and $\mathcal{E}$ represents the set of links among the nodes inside the network.

8.2.1 *DeepWalk*

The DeepWalk [52] algorithm consists of two main components: (1) a random walk generator, and (2) an update procedure. In the first step, the DeepWalk model randomly selects a node, e.g., $u \in \mathcal{V}$, as the root of a random walk W_u from the nodes in the network. Random walk W_u will sample the neighbors of the node last visited uniformly until the maximum length l is met. In the second step, the sampled neighbors are used to update the representations of the nodes inside the graph, where *SkipGram* [46] is applied here.

Random Walk Generator: The random walk model has been introduced in Section *5.1.1*. Formally, the random walk starting at node $u \in \mathcal{V}$ can be represented as W_u, which actually denotes a stochastic process with random status W_u^0, W_u^1, $\cdots$, W_u^k. Formally, at the very beginning, i.e., step 0, the random walk is at the initial node, i.e., $W_u^0 = u$. The status variable W_u^k denotes the node where the node is at step k.

Random walk can capture the local network structures effectively, where the neighborhood and social connection closeness can affect the next nodes that the random walk will move to in the next step. Therefore, in the DeepWalk, random walk is applied to sample a stream of short random walks as the tool for extracting information from a network. Random walk can provide two very desirable properties, besides the ability to capture the local community structures. Firstly, the random walk based local exploration is easy to parallelize. Several random walks can simultaneously explore different parts of the same network in different threads, processes and machines. Secondly, with the information obtained from short random walks, it is possible to accommodate small changes in the network structure without the need for global recomputation.

SkipGram Technique: The updating procedure used in DeepWalk is very similar to the word appearance prediction in language modeling. SkipGram is a language model that maximize the co-occurrence probability of words appearing in the time window s in a sentence. Here, when applying the *SkipGram* technique to the DeepWalk model, the nodes $u \in \mathcal{V}$ in the network can be regarded as the words w denoted in the equations aforementioned. Meanwhile, for the nodes sampled by the random walk model within the window size s before and after node v, they will be treated as the words appearing ahead of and after node v. Furthermore, SkipGram assumes the appearance of the words (or nodes for networks) to be independent, and the above probability equations can be rewritten as follows:

$$P(\{u_{n-s}, u_{n-s+1}, \cdots, u_{n+s}\} \setminus \{u_n\} | \mathbf{x}_{u_n}) = \prod_{i=n-s, i \neq n}^{n+s} P(u_i | \mathbf{x}_{u_n}), \quad (117)$$

where $u_{n-s}, u_{n-s+1}, \cdots, u_{n+s}$ denotes the sequence of nodes sampled by the random walk model.

The learning process of the SkipGram algorithm is provided in Algorithm **??**, where we will enumerate all the co-locations of nodes in the sampled node series u_{n-s}, u_{n-s+1},
$\cdots, u_{n+s}$ by a random walk W_u (starting from node u in the network). With gradient descent, the representation of nodes with their neighbors representations can be updated with stochastic gradient descent. The derivatives are estimated with the back-propagation algorithm. However, in the equation, we need to have the conditional probabilities of the nodes and their representations. A concrete representation of the probability can be a great challenging problem. As proposed in [46], such a distribution can be learnt with some existing models, like logistic regression. However, since the labels used here denote the nodes in the network, it will lead to a very large label space with $|\mathcal{V}|$ different labels, which renders the learning process extremely time consuming. To solve such a problem, some techniques, like Hierarchical Softmax, have been proposed which represents the nodes in the network as a binary tree and can lower done the probability computation time complexity from $O(|\mathcal{V}|)$ to $O(\log |\mathcal{V}|)$.

Hierarchical Softmax: In the SkipGram algorithm, calculating probability $P(u_i|\mathbf{x}_{u_n})$ is infeasible. Therefore, in the DeepWalk model, *hierarchical softmax* is used to factorize the conditional probability. In *hierarchical softmax*, a binary tree is constructed, where the number of leaves equals to the network node set size, and each network node is assigned to a leaf node. The prediction problem is turned into a path probability maximization problem. If a path $(b_0, b_1, \cdots, b_{\lceil \log |\mathcal{V}| \rceil})$ is identified from the tree root to the node u_k, i.e., $b_0 =$ root and $b_{\lceil \log |\mathcal{V}| \rceil} = u_k$, then the probability can be rewritten as

$$P(u_i|\mathbf{x}_{u_n}) = \prod_{l=1}^{\lceil \log |\mathcal{V}| \rceil} P(b_l|\mathbf{x}_{u_n}), \quad (118)$$

where $P(b_l|\mathbf{x}_{u_n})$ can be modeled by a binary classifier denoted as

$$P(b_l|\mathbf{x}_{u_n}) = \frac{1}{1 + e^{-\mathbf{x}_{b_l} \cdot \mathbf{x}_{u_n}}}. \quad (119)$$

Here the parameters involved in the learning process include the representations for both the nodes in the network as well as the nodes in the constructed binary trees.

8.2.2 LINE

To handle the real-world information networks, the embedding models need to have several requirements: (1) preserve the *first-order* and *second-order* proximity between the nodes, (2) scalable to large sized networks, and (3) able to handle networks with different links: *directed* and *undirected*, *weighted* and *unweighted*. In this part, we will introduce another homogeneous network embedding model, named LINE [67].

First-order Proximity: In the network embedding process, the network structure should be effectively preserved, where the node closeness is defined as the node *proximity* concept in LINE. The *first-order proximity* in a network denotes the *local* pairwise proximity between nodes. For a link $(u, v) \in \mathcal{E}$ in the network, the *first-order proximity* denotes the weight of link (u, v) in the network (or 1 if the network is unweighted). Meanwhile, if link (u, v) doesn't exist in the network, the *first-order proximity* between them will be 0 instead. To model the *first-order proximity*, for a given link $(u, v) \in \mathcal{E}$ in the network G, LINE defines the joint probability between nodes u and v as

$$p_1(u, v) = \frac{1}{1 + e^{-\mathbf{x}_u \cdot \mathbf{x}_v}}, \quad (120)$$

where $\mathbf{x}_u, \mathbf{x}_v \in \mathbb{R}^d$ denote the vector representations of nodes u and v respectively.

Function $p_1(\cdot, \cdot)$ defines the proximity distribution in the space of $\mathcal{V} \times \mathcal{V}$. Meanwhile, given a network G, the *empirical proximity* between nodes u and v can be denoted as

$$\hat{p_1}(u, v) = \frac{w_{(u,v)}}{\sum_{(u,v) \in \mathcal{E}} w_{(u,v)}}. \quad (121)$$

To preserve the *first-order proximity*, LINE defines the objective function for the network embedding as

$$J_1 = d(p_1(\cdot, \cdot), \hat{p_1}(\cdot, \cdot)), \quad (122)$$

where function $d(\cdot, \cdot)$ denotes the distance between between the introduced proximity distribution and the empirical proximity distribution. By replacing the distance function $d(\cdot, \cdot)$ with the KL-divergence and omitting some constants, the objective function can be rewritten as

$$J_1 = - \sum_{(u,v) \in \mathcal{E}} w_{(u,v)} \log p_1(u, v). \quad (123)$$

By minimizing the objective function, LINE can learn the feature representation $\mathbf{x}_u$ for each node $u \in \mathcal{V}$ in the network.

Second-order Proximity: In the real-world social networks, the links among the nodes can be very sparse, where the *first-order proximity* can hardly preserve the complete structure information of the network. LINE introduce the concept of *second-order proximity*, where denotes the similarity between the neighborhood structure of nodes. Given a user pair (u, v) in the network, the more common neighbors shared by them, the closer users u and v are in the network. Besides the original representation $\mathbf{x}_u$ for node $u \in \mathcal{V}$, the nodes are also associated with a feature vector representing its context in the network, which is denoted as $\mathbf{y}_u \in \mathbb{R}^d$. Formally, for a given link $(u, v) \in \mathcal{E}$, the probability of context $\mathbf{y}_v$ generated by node u can be represented as

$$p_2(v|u) = \frac{e^{\mathbf{x}_u^\top \cdot \mathbf{y}_v}}{\sum_{v' \in \mathcal{V}} e^{\mathbf{x}_u^\top \cdot \mathbf{y}_{v'}}}. \quad (124)$$

Slightly different from *first-order proximity*, the *second-order* empirical proximity is denoted as

$$\hat{p_2}(v|u) = \frac{w_{(u,v)}}{D(u)}. \tag{125}$$

By minimizing the difference between the introduced proximity distribution and the empirical proximity distribution, the objective function for the *second-order* proximity can be represented as

$$J_2 = \sum_{u \in \mathcal{V}} \lambda_u d(p_2(\cdot|u), \hat{p_2}(\cdot|u)), \tag{126}$$

where λ_u denotes the prestige of node u in the network. Here, by replacing the distance function $d(\cdot|\cdot)$ with the KL-divergence and setting $\lambda_u = D(u)$, the *second-order proximity* based objective function can be represented as

$$J_2 = - \sum_{(u,v) \in \mathcal{E}} w_{(u,v)} \log p_2(v|u). \tag{127}$$

Model Optimization: Instead of combining the *first-order proximity* and *second-order proximity* into a joint optimization function, LINE learns the embedding vectors based on Equations 123 and 127 respectively, which will be further concatenated together to obtain the final embedding vectors.

In optimizing objective function 127, LINE needs to calculate the conditional probability $P(\cdot|u)$ for all nodes $u \in \mathcal{V}$ in the network, which is computational infeasible. To solve the problem, LINE uses the negative sampling approach instead. For each link $(u, v) \in \mathcal{E}$, LINE samples a set of negative links according to some noisy distribution.

Formally, for link $(u, v) \in \mathcal{E}$, the set of negative links sampled for it can be represented as $\mathcal{L}^-_{(u,v)} \subset \mathcal{V} \times \mathcal{V}$. The objective function defined for link (u, v) can be represented as

$$\log \sigma(\mathbf{y}_v^\top \cdot \mathbf{x}_u) + \sum_{(u,v') \in \mathcal{L}^-_{(u,v)}} \log \sigma(-\mathbf{y}_{v'}^\top \cdot \mathbf{x}_u), \tag{128}$$

where $\sigma(\cdot)$ is the sigmoid function. The first term in the above equation denotes the observed links, and the second term represents the negative links drawn from the noisy distribution. Similar approach can also be applied to solve the objective function in Equation 123 as well. The new objective function can be solved with the asynchronous stochastic gradient algorithm (ASGD), which samples a mini-batch of links and then update the parameters.

8.2.3 node2vec

In LINE, the closeness among nodes in the networks is preserved based on either the *first-order proximity* or the *second-order proximity*. In a recent work, node2vec [22], the authors propose to preserve the proximity between nodes with a sampled set of nodes in the network.

node2vec Framework: Model node2vec is based on the *SkipGram* in language modeling, and the objective function of node2vec can be formally represented as

$$\max \sum_{u \in \mathcal{V}} \log P(\Gamma(u)|\mathbf{x}_u). \tag{129}$$

where $\mathbf{x}_u$ denotes the latent feature vector learnt for node u and $\Gamma(u)$ represents the neighbor set of node u in the network.

To simplify the problem and make the problem solvable, some assumptions are made to approximate the objective function into a simpler form.

- *Conditional Independence Assumption*: Given the latent feature vector $\mathbf{x}_u$ of node u, by assuming the observation of node in set $\Gamma(u)$ to be independent, the probability equation can be rewritten as

$$P(\Gamma(u)|\mathbf{x}_u) = \prod_{v \in \Gamma(u)} P(v|\mathbf{x}_u). \tag{130}$$

- *Symmetric Node Effect*: Furthermore, by assuming the source and neighbor nodes have a symmetric effect on each other in the feature space, the conditional probability $P(v|\mathbf{x}_u)$ can be rewritten as

$$P(v|\mathbf{x}_u) = \frac{e^{\mathbf{x}_v^\top \cdot \mathbf{x}_u}}{\sum_{v' \in \mathcal{V}} e^{\mathbf{x}_{v'}^\top} \cdot \mathbf{x}_u}. \tag{131}$$

Therefore, the objective function can be simplified as

$$\max_{\mathbf{X}} \sum_{u \in \mathcal{V}} [-\log Z_u + \sum_{v' \in \Gamma(u)} \mathbf{x}_{v'}^\top \cdot \mathbf{x}_u], \tag{132}$$

where $Z_u = \sum_{v' \in \mathcal{V}} e^{\mathbf{x}_{v'}^\top \cdot \mathbf{x}_u}$. Term Z_u will be different for different nodes $u \in \mathcal{V}$, which is expensive to compute for large networks, and node2vec proposes to apply the negative sampling technique instead. The main issue discussed in node2vec is about sampling the neighborhood set $\Gamma(u)$ from the network.

To overcome the shortcomings of BFS and DFS, node2vec proposes to apply random walk to sample the neighborhood set $\Gamma(u)$ instead. Given a random walk W, the node W resides at in step i can be represented as variable $s_i \in \mathcal{V}$. The complete sequence of nodes that W has resides at can be represented as $s_0, s_1, \cdots, s_k$, where s_0 denotes the initial node starting the walk. The transitional probability from node u to v in W in the i_{th} step can be denoted as

$$P(s_i = v|s_{i-1} = u) = \begin{cases} w_{(u,v)} & \text{if } (u,v) \in \mathcal{E}, \\ 0, & \text{otherwise}, \end{cases} \tag{133}$$

where $w_{(u,v)}$ denotes the normalized weight of link (u, v) in the network ($w_{(u,v)} = 1$ if the network is unweighted).

Traditional random walk model doesn't take account for the network structure and can hardly explore different network neighborhoods. node2vec adapts the random walk model and introduce the 2_{nd} order random walk model with parameters p and q, which will help guide the walk. In node2vec, let's assume the walk just traversed link (t, u) and can go to node v in the next step. Formally, the transitional probability of link (u, v) is adjusted with parameter $\alpha_{p,q}(t, v)$ (i.e., $w_{(u,v)} = \alpha_{p,q}(t, v) \cdot w_{(u,v)}$), where

$$\alpha_{p,q}(t,v) = \begin{cases} \frac{1}{p}, & \text{if } d_{t,v} = 0, \\ 1, & \text{if } d_{t,v} = 1, \\ \frac{1}{q}, & \text{if } d_{t,v} = 2, \end{cases} \tag{134}$$

where $d_{t,v}$ denotes the shortest distance between nodes t and v in the network. Since the walk can go from t to u, and then from u to v, the distance from t to v will be at most 2.

Parameters p and q control the walk transition sequence effectively, where parameter p is also called the *return parameter* and q is called the *in-out parameter* in node2vec.

8.3 Emerging Network Embedding

We have introduce several network embedding models in the previous sections already. However, when applied to handle real-world social network data, these existing embedding models can hardly work well. The main reason is that the network internal social links are usually very sparse in online soical networks [67], which can hardly preserve the complete network structure. For a pair of users

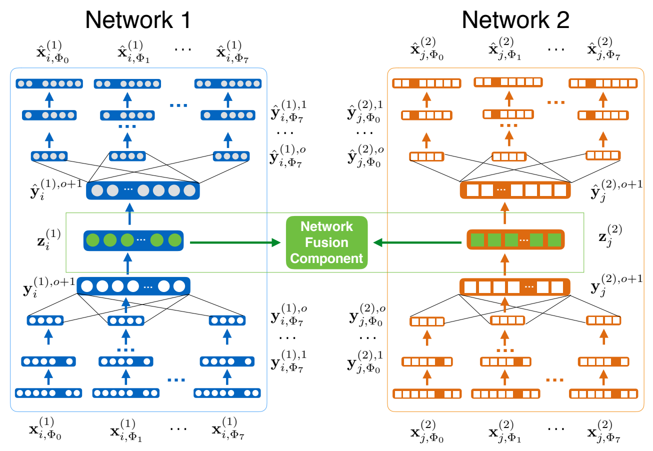

Figure 4: The DIME Framework.

who are not directed connected, these models will not be able determine the closeness of these users' feature vectors in the embedding space. Such a problem will be more severe when it comes to the *emerging social networks* [95], which denote the newly created online social networks containing very few social connections.

In this section, we will study the emerging network embedding problem across multiple aligned heterogeneous social networks simultaneously. To solve the problem, in this section, we will introduce a novel aligned heterogeneous social network embedding framework, named DIME proposed in [93].

8.3.1 *Deep* DIME-SH *Model*

DIME is based on the *aligned auto-encoder model*, which extends the traditional *deep auto-encoder model* to the *multiple aligned heterogeneous networks* scenario. In the application of DIME on a heterogeneous network $G^{(1)}$, multiple node *meta proximity* matrices will be extracted (i.e., $\{\mathbf{P}^{(1)}_{\Phi_0}, \mathbf{P}^{(1)}_{\Phi_1}, \cdots, \mathbf{P}^{(1)}_{\Phi_7}\}$) based on the *intra-network meta paths*. As shown in the architecture in Figure 4 (either the left component for network 1 or the right component for network 2), about the same instance, DIME-SH (DIME-Single Heterogeneous Network) takes different feature vectors extracted from the meta paths $\{\Phi_0, \Phi_1, \cdots, \Phi_7\}$ as the input. For each meta path, a series of separated encoder and decoder steps are carried out simultaneously, whose latent vectors are fused together to calculate the final embedding vector $\mathbf{z}^{(1)}_i \in \mathbb{R}^{d^{(1)}}$ for user $u^{(1)}_i \in \mathcal{V}^{(1)}$. In the DIME-SH model, the input feature vectors (based on meta path $\Phi_k \in \{\Phi_0, \Phi_1, \cdots, \Phi_7\}$) of user u_i can be represented as $\mathbf{x}^{(1)}_{i,\Phi_k}$, which denotes the row corresponding to users $u^{(1)}_i$ in matrix $\mathbf{P}^{(1)}_{\Phi_k}$ defined before. Meanwhile, the latent representation of the instance based on the feature vector extracted via meta path Φ_k at different hidden layers can be represented as $\{\mathbf{y}^{(1),1}_{i,\Phi_k}, \mathbf{y}^{(1),2}_{i,\Phi_k}, \cdots, \mathbf{y}^{(1),o}_{i,\Phi_k}\}$.

One of the significant difference of model DIME-SH from traditional auto-encoder model lies in the (1) combination of various hidden vectors $\{\mathbf{y}^{(1),o}_{i,\Phi_0}, \mathbf{y}^{(1),o}_{i,\Phi_1}, \cdots, \mathbf{y}^{(1),o}_{i,\Phi_7}\}$ to obtain the final embedding vector $\mathbf{z}^{(1)}_i$ in the encoder step, and (2) the dispatch of the embedding vector $\mathbf{z}^{(1)}_i$ back to the hidden vectors in the decoder step. As shown in the architecture, formally, these extra steps can be represented as

$$\begin{cases} \text{\# extra encoder steps} \\ \mathbf{y}^{(1),o+1}_i = \sigma(\sum_{\Phi_k \in \{\Phi_0, \cdots, \Phi_7\}} \mathbf{W}^{(1),o+1}_{\Phi_k} \mathbf{y}^{(1),o}_{i,\Phi_k} + \mathbf{b}^{(1),o+1}_{\Phi_k}), \\ \mathbf{z}^{(1)}_i = \sigma(\mathbf{W}^{(1),o+2} \mathbf{y}^{(1),o+1}_i + \mathbf{b}^{(1),o+2}). \\ \text{\# extra decoder steps} \\ \hat{\mathbf{y}}^{(1),o+1}_i = \sigma(\hat{\mathbf{W}}^{(1),o+2} \mathbf{z}^{(1)}_i + \hat{\mathbf{b}}^{(1),o+2}), \\ \hat{\mathbf{y}}^{(1),o}_{i,\Phi_k} = \sigma(\hat{\mathbf{W}}^{(1),o+1}_{\Phi_k} \hat{\mathbf{y}}^{(1),o+1}_i + \hat{\mathbf{b}}^{(1),o+1}_{\Phi_k}). \end{cases} \quad (135)$$

What's more, since the input feature vectors are extremely sparse (lots of the entries have value 0s), simply feeding them to the model may lead to some trivial solutions, like $\mathbf{0}$ vectors for both $\mathbf{z}^{(1)}_i$ and the decoded vectors $\hat{\mathbf{x}}^{(1)}_{i,\Phi_k}$. To overcome such a problem, another significant difference of model DIME-SH from traditional auto-encoder model lies in the loss function definition, where the loss introduced by the non-zero features will be assigned with a larger weight. In addition, by adding the loss function for each of the meta paths, the final loss function in DIME-SH can be formally represented as

$$\mathcal{L}^{(1)} = \sum_{\Phi_k \in \{\Phi_0, \cdots, \Phi_7\}} \sum_{u_i \in \mathcal{V}} \left\| \left(\mathbf{x}^{(1)}_{i,\Phi_k} - \hat{\mathbf{x}}^{(1)}_{i,\Phi_k} \right) \odot \mathbf{b}^{(1)}_{i,\Phi_k} \right\|_2^2, \quad (136)$$

where vector $\mathbf{b}^{(1)}_{i,\Phi_k}$ is the weight vector corresponding to feature vector $\mathbf{x}^{(1)}_{i,\Phi_k}$. Entries in vector $\mathbf{b}^{(1)}_{i,\Phi_k}$ are filled with value 1s except the entries corresponding to non-zero element in $\mathbf{x}^{(1)}_{i,\Phi_k}$, which will be assigned with value γ ($\gamma > 1$ denoting a larger weight to fit these features). In a similar way, the loss function for the embedding result in network $G^{(2)}$ can be formally represented as $\mathcal{L}^{(2)}$.

8.3.2 *Deep* DIME *Framework*

By accommodating the embedding between the aligned networks, information can be transferred from the aligned mature network to refine the embedding result in the emerging network effectively. The complete architecture of DIME is shown in Figure 4, which involve the DIME-SH components for each of the aligned networks, where the information transfer component aligns these separated DIME-SH models together. To be more specific, given a pair of aligned heterogeneous networks $\mathcal{G} = ((G^{(1)}, G^{(2)}), \mathcal{A}^{(1,2)})$ ($G^{(1)}$ is an emerging network and $G^{(2)}$ is a mature network), the embedding results can be represented as matrices $\mathbf{Z}^{(1)} \in \mathbb{R}^{|\mathcal{U}^{(1)}| \times d^{(1)}}$ and $\mathbf{Z}^{(2)} \in \mathbb{R}^{|\mathcal{U}^{(2)}| \times d^{(2)}}$ for all the user nodes in $G^{(1)}$ and $G^{(2)}$ respectively. The i_{th} row of matrix $\mathbf{Z}^{(1)}$ (or the j_{th} row of matrix $\mathbf{Z}^{(2)}$) denotes the encoded feature vector of user $u^{(1)}_i$ in $G^{(1)}$ (or $u^{(2)}_j$ in $G^{(2)}$). If $u^{(1)}_i$ and $u^{(2)}_j$ are the same user, i.e., $(u^{(1)}_i, u^{(2)}_j) \in \mathcal{A}^{(1,2)}$, by placing vectors $\mathbf{Z}^{(1)}(i,:)$ and $\mathbf{Z}^{(2)}(j,:)$ in a close region in the embedding space, information from $G^{(2)}$ can be used to refine the embedding result in $G^{(1)}$.

Information transfer is achieved based on the anchor links, and we only care about the anchor users. To adjust the rows of matrices $\mathbf{Z}^{(1)}$ and $\mathbf{Z}^{(2)}$ to remove non-anchor users and make the same rows correspond to the same user, DIME introduces the binary inter-network transitional matrix $\mathbf{T}^{(1,2)} \in \mathbb{R}^{|\mathcal{U}^{(1)}| \times |\mathcal{U}^{(2)}|}$. Entry $T^{(1,2)}(i,j) = 1$ iff the corresponding users are connected by anchor links, i.e., $(u^{(1)}_i, u^{(2)}_j) \in \mathcal{A}^{(1,2)}$. Furthermore, the encoded feature vectors for users in these two networks can be of different dimensions, i.e., $d^{(1)} \neq d^{(2)}$, which can be accommodated via the projection $\mathbf{W}^{(1,2)} \in \mathbb{R}^{d^{(1)} \times d^{(2)}}$.

Formally, the introduced *information fusion loss* between networks $G^{(1)}$ and $G^{(2)}$ can be represented as

$$\mathcal{L}^{(1,2)} = \left\| (\mathbf{T}^{(1,2)})^\top \mathbf{Z}^{(1)} \mathbf{W}^{(1,2)} - \mathbf{Z}^{(2)} \right\|_F^2. \quad (137)$$

The complete objective function of framework include the loss terms introduced by the component DIME-SH for networks $G^{(1)}$, $G^{(2)}$, and the *information fusion loss*. The latent embedding vectors achieved via DIME-SH on network $G^{(1)}$ define the embedding vectors of user nodes in the emerging network.

9. CONCLUSION AND FUTURE POTENTIAL DEVELOPMENTS

In this paper, we have introduced the current research works on broad learning and its applications on social media studies. This paper has covered 5 main research directions about broad learning based social media studies: (1) *network alignment*, (2) *link prediction*, (3) *community detection*, (4) *information diffusion* and (5) *network embedding*. These problems introduced in this chapter are all very important for many concrete real-world social network applications and services. A number of nontrivial algorithms have been proposed to resolve these problems, which have been talked about in great detail in this paper respectively.

Both the *broad learning* and *social media mining* are very promising research directions, and some potential future development directions are illustrated as follows.

1. **Scalable Broad Learning Algorithms**: Data generated nowadays is usually of very large scale, and fusion of such big data from multiple sources together will render the problem more challenging. For instance, the online social networks (like Facebook) usually involve millions even billions of active users, and the social data generated by these users in each day will consume more than 600 TB storage space (in Facebook). One of the major future development about the *broad learning based social media mining* is to develop scalable data fusion and mining algorithms that can handle such a **large volume** (of **big data**) challenge. One tentative approach is to develop information fusion algorithms based on distributed platforms, like Spark and Hadoop [26], and handle the data with a large distributed computing cluster. Another method to resolve the scalability challenge is from the model optimization perspective. Optimizing existing learning models and proposing new approximated learning algorithms with lower time complexity are desirable in the future research projects. In addition, applications of the latest deep learning models to fuse and mine the large-scale datasets can be another alternative approach for the scalable *broad learning* on social networks.
2. **Multiple Sources Fusion and Mining**: Current research works on multiple source data fusion and mining mainly focus on aligning entities in one single pair of data sources (i.e., two sources), where information exchange between the sources mainly rely on the anchor links between these aligned entities. Meanwhile, when it comes to fusion and mining of multiple (more than two) sources, the problem setting will be quite different and become more challenging. For example, in the alignment of more networks, the transitivity property of the inferred anchor links needs to be preserved [98]. Meanwhile, in the information transfer from multiple external aligned sources to the target source, the information sources should be weighted differently according to their importance. Therefore, the **diverse variety** of the multiple sources will lead to more research challenges and opportunities, which is also a great challenge in **big data** studies. New information fusion and mining algorithms for the multi-source scenarios can be another great opportunity to explore broad learning in the future.
3. **Broader Learning Applications**: Besides the research works on social network datasets, the third potential future development of broad learning and mining lies its broader applications on various categories of datasets, like enterprise internal data [100; 88; 103; 102], geo-spatial data [89; 77; 90], knowledge base data, and pure text data. Some prior research works on fusing enterprise context information sources, like enterprise social networks, organizational chart and employee profile information have been done already [100; 88; 103; 102]. Several interesting problems, like organizational chart inference [100], enterprise link prediction [88], information diffusion at workplace [103] and enterprise employee training [102], have been studied based on the fused enterprise internal information. In the future, these areas are still open for exploration. Applications of broad learning techniques in other application problems, such as employee training, expert location and project team formation, will be both interesting problems awaiting for further investigation. In addition, analysis of the correlation of different traveling modalities (like shared bicycles [89; 77; 90], bus and metro train) with the city zonings in smart city; and fusing multiple knowledge bases, like Douban and IMDB, for knowledge discovery and truth finding are both good application scenarios for broad learning research works.

10. REFERENCES

[1] L. Adamic and E. Adar. Friends and neighbors on the web. *Social Networks*, 2001.

[2] C. Aggarwal. *Data Mining: The Textbook*. Springer Publishing Company, 2015.

[3] A. Arenas, L. Danon, A. Díaz-Guilera, P. M. Gleiser, and R. Guimerá. Community analysis in social networks. *The European Physical Journal B*, 2004.

[4] L. Backstrom and J. Leskovec. Supervised random walks: predicting and recommending links in social networks. In *WSDM*, 2011.

[5] A.-L. Barabasi, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaboration. In *Physica A*, 2002.

[6] M. Bayati, M. Gerritsen, D. Gleich, A. Saberi, and Y. Wang. Algorithms for large, sparse network alignment problems. In *ICDM*, 2009.

[7] S. Bharathi, D. Kempe, and M. Salek. Competitive influence maximization in social networks. In *WINE*, 2007.

[8] T. Blomberg. *Heat conduction in two and three dimensions : computer modelling of building physics applications*. PhD thesis, 1996.

[9] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*. 2013.

[10] N. Chen. On the approximability of influence in social networks. In *SODA*, 2008.

[11] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *IJPRAI*, 2004.

[12] R. Dasgupta, B. Garcia, and R. Goodman. Systemic spread of an rna insect virus in plants expressing plant viral movement protein genes. *Proceedings of the National Academy of Sciences*, 2001.

[13] S. Datta, A. Majumder, and N. Shrivastava. Viral marketing for multiple products. In *ICDM*, 2010.

[14] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Ontology matching: A machine learning approach. In *Handbook on Ontologies*. 2004.

[15] C. Elkan and K. Noto. Learning classifiers from only positive and unlabeled data. In *KDD*, 2008.

[16] M. Eslami, A. Aleyasen, R. Moghaddam, and K. Karahalios. Friend grouping algorithms for online social networks: Preference, bias, and implications. In *Social Informatics*, 2014.

[17] J. Flannick, A. Novak, B. Srinivasan, H. McAdams, and S. Batzoglou. Graemlin: general and robust alignment of multiple large interaction networks. *Genome research*, 2006.

[18] S. Fortin. The graph isomorphism problem. Technical report, 1996.

[19] F. Fouss, A. Pirotte, J. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *TKDE*, 2007.

[20] R. Ghosh, K. Lerman, T. Surachawala, K. Voevodski, and S. Teng. Non-conservative diffusion and its application to social network analysis. *CoRR*, abs/1102.4639, 2011.

[21] D. Gibson, J. Kleinberg, and P. Raghavan. Inferring web communities from link topology. In *HYPERTEXT*, 1998.

[22] A. Grover and J. Leskovec. Node2vec: Scalable feature learning for networks. In *KDD*, 2016.

[23] M. Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In *SDM*, 2006.

[24] M. Hasan and M. Zaki. In *Social Network Data Analytics*. 2011.

[25] Q. Hu, S. Xie, J. Zhang, Q. Zhu, S. Guo, and P. Yu. Heterosales: Utilizing heterogeneous social networks to identify the next enterprise customer. In *WWW*, 2016.

[26] S. Jin, J. Zhang, P. Yu, S. Yang, and A. Li. Synergistic partitioning in multiple large scale social networks. In *BigData*, 2014.

[27] M. Kalaev, V. Bafna, and R. Sharan. Fast and accurate alignment of multiple protein networks. In *RECOMB*. 2008.

[28] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 1953.

[29] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.

[30] D. Kempe, J. Kleinberg, and É. Tardos. Influential nodes in a diffusion model for social networks. In *ICALP*, 2005.

[31] J. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 1999.

[32] X. Kong, J. Zhang, and P. Yu. Inferring anchor links across multiple heterogeneous social networks. In *CIKM*, 2013.

[33] I. Konstas, V. Stathopoulos, and J. M. Jose. On social networks and collaborative recommendation. In *SIGIR*, 2009.

[34] J. Kostka, Y. Oswald, and R. Wattenhofer. Word of mouth: Rumor dissemination in social networks. In *SIROCCO*, 2008.

[35] D. Koutra, H. Tong, and D. Lubensky. Big-align: Fast bipartite graph alignment. In *ICDM*, 2013.

[36] J. Lee, W. Han, R. Kasperovics, and J. Lee. An in-depth comparison of subgraph isomorphism algorithms in graph databases. *VLDB*, 2012.

[37] J. Leskovec, K. Lang, and M. Mahoney. Empirical comparison of algorithms for network community detection. In *WWW*, 2010.

[38] C. Liao, K. Lu, M. Baym, R. Singh, and B. Berger. Isorankn: spectral methods for global alignment of multiple protein networks. *Bioinformatics*, 2009.

[39] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.*, 2007.

[40] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, 2015.

[41] B. Liu, Y. Dai, X. Li, W. Lee, and P. Yu. Building text classifiers using positive and unlabeled examples. In *ICDM*, 2003.

[42] L. Lü and T. Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 2011.

[43] F. D. Malliaros and M. Vazirgiannis. Clustering and community detection in directed networks: A survey. *CoRR*, abs/1308.0971, abs/1308.0971, 2013.

[44] F. Manne and M. Halappanavar. New effective multithreaded matching algorithms. In *IPDPS*, 2014.

[45] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, 2002.

[46] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[47] T. N. Narasimhan. Fourier's heat conduction equation: History, influence, and connections. *Proceedings of the Indian Academy of Sciences - Earth and Planetary Sciences*, 1999.

[48] M. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 2006.

[49] I NÍsell. Stochastic models of some endemic infections. *Mathematical Biosciences*, 2002.

[50] D. Park, R. Singh, M. Baym, C. Liao, and B. Berger. Isobase: a database of functionally related proteins across ppi networks. *Nucleic Acids Research*, 2011.

[51] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, and A. Vespignani. Epidemic processes in complex networks. *Rev. Mod. Phys.*, 2015.

[52] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *KDD*, 2014.

[53] Baron RC, McCormick JB, and Zubeir OA. Ebola virus disease in southern sudan: hospital dissemination and intrafamilial spread. *Bull World Health Organ.*, 1983.

[54] R. Read and D. Corneil. The graph isomorphism disease. 2006.

[55] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD*, 2002.

[56] R. Roman. Community-based recommendations to improve intranet users' productivity. Master's thesis, 2016.

[57] W. Shao, J. Zhang, L. He, and P. Yu. Multi-source multi-view clustering via discrepancy penalty. In *IJCNN*, 2016.

[58] R. Sharan, S. Suthram, R. Kelley, T. Kuhn, S. McCuine, P. Uetz, T. Sittler, R. Karp, and T. Ideker. Conserved patterns of protein interaction in multiple species. 2005.

[59] J. Shi and J. Malik. Normalized cuts and image segmentation. *TPAMI*, 2000.

[60] Y. Shih and S. Parthasarathy. Scalable global alignment for multiple biological networks. *Bioinformatics*, 2012.

[61] R. Singh, J. Xu, and B. Berger. Pairwise global alignment of protein interaction networks by matching neighborhood topology. In *RECOMB*, 2007.

[62] R. Singh, J. Xu, and B. Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences*, 2008.

[63] A. Smalter, J. Huan, and G. Lushington. Gpm: A graph pattern matching kernel with diffusion for chemical compound classification. In *IEEE BIBE*, 2008.

[64] B. Sriperumbudur and G. Lanckriet. On the convergence of concave-convex procedure. In *NIPS*, 2009.

[65] Y. Sun, J. Han, X. Yan, P. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB*, 2011.

[66] Y. Sun, Y. Yu, and J. Han. Ranking-based clustering of heterogeneous information networks with star network schema. In *KDD*, 2009.

[67] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *WWW*, 2015.

[68] H. Tong, C. Faloutsos, and J. Pan. Fast random walk with restart and its applications. In *ICDM*, 2006.

[69] M. Trusov, A. Bodapati, and R. Bucklin. Determining Influential Users in Internet Social Networks. *Journal of Marketing Research*, 2010.

[70] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE TPAMI*, 1988.

[71] U. von Luxburg. A tutorial on spectral clustering. *CoRR*, 2007.

[72] X. Wang and G. Chen. Complex networks: small-world, scale-free and beyond. *IEEE Circuits and Systems Magazine*, 2003.

[73] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, 2014.

[74] J. Yang, J. McAuley, and J. Leskovec. Community detection in networks with node attributes. In *ICDM*, 2013.

[75] A. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 2003.

[76] R. Zafarani and H. Liu. Connecting users across social media sites: A behavioral-modeling approach. In *KDD*, 2013.

[77] Q. Zhan, J. Zhang, X. Pan, M. Li, and P. Yu. Discover tipping users for cross network influencing. In *IRI*, 2016.

[78] Q. Zhan, J. Zhang, S. Wang, P. Yu, and J. Xie. Influence maximization across partially aligned heterogenous social networks. In *PAKDD*, 2015.

[79] Q. Zhan, J. Zhang, P. Yu, S. Emery, and J. Xie. Inferring social influence of anti-tobacco mass media campaigns. In *BIBM*, 2016.

[80] Q. Zhan, J. Zhang, P. Yu, and J. Xie. Viral marketing through aligned networks. 2017.

[81] H. Zhang, D. Nguyen, S. Das, H. Zhang, and M. Thai. Least cost influence maximization across multiple social networks. *CoRR*, abs/1606.08927, 2016.

[82] J. Zhang, C. Aggarwal, and P. Yu. Rumor initiator detection in infected signed networks. In *ICDCS*, 2017.

[83] J. Zhang, J. Chen, S. Zhi, Y. Chang, P. Yu, and J. Han. Link prediction across aligned networks with sparse low rank matrix estimation. In *ICDE*, 2017.

[84] J. Zhang, J. Chen, J. Zhu, Y. Chang, and P. Yu. Link prediction with cardinality constraints. In *WSDM*, 2017.

[85] J. Zhang, L. Cui, P. Yu, and Y. Lv. Bl-ecd: Broad learning based enterprise community detection via hierarchical structure fusion. In *CIKM*, 2017.

[86] J. Zhang, X. Kong, and P. Yu. Predicting social links for new users across aligned heterogeneous social networks. In *ICDM*, 2013.

[87] J. Zhang, X. Kong, and P. Yu. Transferring heterogeneous links across location-based social networks. In *WSDM*, 2014.

[88] J. Zhang, Y. Lv, and P. Yu. Enterprise social link prediction. In *CIKM*, 2015.

[89] J. Zhang, X. Pan, M. Li, and P. Yu. Bicycle-sharing system analysis and trip prediction. In *MDM*, 2016.

[90] J. Zhang, X. Pan, M. Li, and P. Yu. Bicycle-sharing systems expansion: Station re-deployment through crowd planning. In *SIGSPATIAL*, 2016.

[91] J. Zhang, W. Shao, S. Wang, X. Kong, and P. Yu. Pna: Partial network alignment with generic stable matching. In *IRI*, 2015.

[92] J. Zhang, S. Wang, Q. Zhan, and P. Yu. Intertwined viral marketing in social networks. In *ASONAM*, 2016.

[93] J. Zhang, C. Xia, C. Zhang, L. Cui, Y. Fu, and P. Yu. Bl-mne: Emerging heterogeneous social network embedding through broad learning with aligned autoencoder. In *ICDM*, 2017.

[94] J. Zhang and P. Yu. Link prediction across heterogeneous social networks: A survey. 2014.

[95] J. Zhang and P. Yu. Community detection for emerging networks. In *SDM*, 2015.

[96] J. Zhang and P. Yu. Integrated anchor and social link predictions across partially aligned social networks. In *IJCAI*, 2015.

[97] J. Zhang and P. Yu. Mcd: Mutual clustering across multiple social networks. In *BigData Congress*, 2015.

[98] J. Zhang and P. Yu. Multiple anonymized social networks alignment. In *ICDM*, 2015.

[99] J. Zhang and P. Yu. Pct: Partial co-alignment of social networks. In *WWW*, 2016.

[100] J. Zhang, P. Yu, and Y. Lv. Organizational chart inference. In *KDD*, 2015.

[101] J. Zhang, P. Yu, and Y. Lv. Enterprise community detection. In *ICDE*, 2017.

[102] J. Zhang, P. Yu, and Y. Lv. Enterprise employee training via project team formation. In *WSDM*, 2017.

[103] J. Zhang, P. Yu, Y. Lv, and Q. Zhan. Information diffusion at workplace. In *CIKM*, 2016.

[104] J. Zhang, P. Yu, and Z. Zhou. Meta-path based multi-network collective link prediction. In *KDD*, 2014.

[105] Y. Zhang and D. Yeung. Overlapping community detection via bounded nonnegative matrix tri-factorization. In *KDD*, 2012.

[106] Y. Zhao, E. Levina, and J. Zhu. Community extraction for social networks. *Proceedings of the National Academy of Sciences*, 2011.

[107] T. Zhou, L. Lü, and Y. Zhang. Predicting missing links via local information. *The European Physical Journal B*, 2009.

[108] J. Zhu, J. Zhang, L. He, Q. Wu, B. Zhou, C. Zhang, and P. Yu. Broad learning based multi-source collaborative recommendation. In *CIKM*, 2017.