BL-MNE: Emerging Heterogeneous Social Network Embedding through Broad Learning with Aligned Autoencoder

Jiawei Zhang⋆, Congying Xia§, Chenwei Zhang§, Limeng Cui†, Yanjie Fu‡ and Philip S. Yu§,¶

⋆IFM Lab, Department of Computer Science, Florida State University, FL, USA
§University of Illinois at Chicago, Chicago, IL, USA
†School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing, China
‡Missouri University of Science and Technology, MO, USA
¶Shanghai Institute for Advanced Communication and Data Science, Fudan University, Shanghai, China
jzhang@cs.fsu.edu, {cxia8, czhang99, psyu}@uic.edu, lmcui932@163.com, fuyan@mst.edu

Abstract—Network embedding aims at projecting the network data into a low-dimensional feature space, where the nodes are represented as a unique feature vector and network structure can be effectively preserved. In recent years, more and more online application service sites can be represented as massive and complex networks, which are extremely challenging for traditional machine learning algorithms to deal with. Effective embedding of the complex network data into low-dimension feature representation can both save data storage space and enable traditional machine learning algorithms applicable to handle the network data. Network embedding performance will degrade greatly if the networks are of a sparse structure, like the emerging networks with few connections. In this paper, we propose to learn the embedding representation for a target emerging network based on the broad learning setting, where the emerging network is aligned with other external mature networks at the same time. To solve the problem, a new embedding framework, namely "Deep alIgned autoencoder based eMbEdding" (DIME), is introduced in this paper. DIME handles the diverse link and attribute in a unified analytic based on broad learning, and introduces the multiple aligned attributed heterogeneous social network concept to model the network structure. A set of meta paths are introduced in the paper, which define various kinds of connections among users via the heterogeneous link and attribute information. The closeness among users in the networks are defined as the meta proximity scores, which will be fed into DIME to learn the embedding vectors of users in the emerging network. Extensive experiments have been done on real-world aligned social networks, which have demonstrated the effectiveness of DIME in learning the emerging network embedding vectors.

I. INTRODUCTION

In the era of big data, a rapidly increasing number of online application websites appear recently, which can be represented as massive and complex networks. The representative examples include *online social networks*, like Facebook and Twitter, *e-commerce sites*, like Amazon and eBay, *academic sites*, like DBLP and Google Scholar, as well as *POIs recommendation sites*, like Foursquare and Yelp. These network data can be very difficult to deal with due to their *extremely large scale* (involving millions even billions of nodes), *complex structures* (containing heterogeneous links) as well as *diverse attributes* (attached to the nodes or links). Great challenges exist in handling these complex network representation data with traditional machine learning algorithms, which usually take feature vectors as the input and cannot handle graph data directly. A general representation of heterogeneous networks as feature vectors is desired for knowledge discovery from such complex-structured data. In this paper, we will use online social networks as the example to illustrate the studied problem as well as the learning framework.

In recent years, many research works propose to embed the online social network data into a low-dimensional feature space [14], [18], [6], in which each node is represented as a unique feature vector. From these feature vectors, the original network structure can be effectively reconstructed. With these embedded feature vectors,

classic machine learning algorithms can be applied to deal with the social network data directly, and the storage space can also be saved greatly. However, most existing social network embedding methods are proposed for homogeneous networks, which learn the feature vectors for user nodes merely based on the social connections among them. When applied to handle real-world social network data, these embedding models can hardly work well. The main reason is that the internal social links are usually very sparse in online social networks [18], which can hardly preserve the complete network structure. For a pair of users 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* [24], which denote the newly created online social networks with very few social connections.

Meanwhile, as discovered in [30], to enjoy more social network services, people nowadays are usually involved in multiple online social networks at the same time. For instance, people tend to join in Facebook for casual socialization with their classmates; they will use Foursquare to search for nearby restaurants for dinner; and they will turn to use Instagram to share photos with their friends online. Users who are involved in these emerging social networks may have been using other well-developed social networks (e.g., Facebook, Twitter) for a long time. Information available for the users in other aligned mature networks is usually very abundant and of a sufficient amount. Effective information exchanges from these mature networks to the emerging networks for the shared users can help overcome the information sparsity problem promisingly, which is a important topic covered in the broad learning task [22], [31] to be introduced as follows. To denote the accounts owned by the same people in different online social networks, an *anchor link* will be added to connect their account pair between the networks [30]. Formally, the online social networks connected by the anchor links (between the shared user accounts) are called *multiple aligned social networks* [30].

Problem Studied: In this paper, we propose to study the emerging network embedding problem across multiple aligned heterogeneous social networks simultaneously based on the broad learning setting, which is formally named as the "Broad Learning based eMerging Network Embedding" (BL-MNE) problem. In the concurrent embedding process based on the broad learning setting, BL-MNE aims at distilling relevant information from both the emerging and other aligned mature networks to derive compliment knowledge and learn a good vector representation for user nodes in the emerging network.

Here, "**Broad Learning**" [22], [31] 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 [25], [21],

TABLE I
Summary of related problems.

Property	Aligned Heterogeneous Network Embedding	Translation based Graph Embedding [3], [19], [11]	Homogeneous Network Embedding [14], [18], [6]	Heterogeneous Network Embedding [4], [5]
target network	emerging	regular	regular	regular
network	attributed heterogeneous	multi-relational	homogeneous	heterogeneous
#network	multiple	single	single	single
proximity	meta proximity	first order	first/second order/random walk	first order [4], meta path [5]
multi-source fusion	anchor link based fusion	N/A	N/A	N/A

[30], [24], [26], [23], [29], [27], [28]. In the real world, on the same information entities, e.g., social media users [25], [21], [30], [24], movie knowledge library entries [31] and employees in companies [26], [23], [29], [27], [28], a large amount of information can actually be collected from various sources. These sources are usually of different varieties, like Foursquare vs Twitter [25], [21], [30], [24], IMDB vs Douban Movie sites [31], Yammer vs company organizational chart [26], [23], [29], [27], [28]. Each information source provides a specific signature of the same entity from a unique underlying aspect. Effective fusion of these different information sources provides an opportunity for researchers and practitioners to understand the entities more comprehensively, which renders "**Broad Learning**" an extremely important learning task. 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 information sources, and evaluates the corresponding benefits [22], [31]. Great challenges exist in "**Broad Learning**" for the effective fusion of relevant knowledge across different aligned information sources depends upon not only the relatedness of these information sources, but also the target application problems. "**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 [22], [31].

BL-MNE is significantly different from existing network embedding problems [3], [19], [14], [11], [18], [4], [6], [5] in several perspectives. First of all, the target network studied in BL-MNE is an emerging network suffering from the information sparsity problem, which is different from the embedding problems for regular networks [3], [19], [11], [4], [5]. Secondly, the networks studied in BL-MNE are all heterogeneous networks containing complex links and diverse attributes, which renders BL-MNE different from existing homogeneous network embedding problems [14], [18], [6]. Furthermore, BL-MNE is based on the multiple aligned networks setting, where information from aligned networks will be exchanged to refine the embedding results mutually, and it is different from the existing single-network based embedding problems [3], [19], [14], [11], [4], [18], [4], [6], [5]. We also provide a summary about the difference between BL-MNE and existing works in Table I (which summarizes and compares several related works in different aspects), and more information about other related works will be introduced Section V at the end of the paper.

The BL-MNE problem is not an easy problem, and it has several great challenges to deal with, which are provided as follows:

- *Problem Formulation*: To overcome the information sparsity problem, BL-MNE studies the concurrent embedding of multiple aligned social networks, which is still an open problem to this context so far. Formal definition and formulation of the BL-MNE problem is required before we introduce the solutions.
- *Heterogeneity of Networks*: The networks studied in this paper are of very complex structures. Besides the regular social connections among users, there also exist many other types of links as well as diverse attributes attached to the user nodes.

Effective incorporating these heterogeneous information into a unified embedding analytic is a great challenge.

- *Multiple Aligned Network Embedding Framework*: Due to the significant differences between BL-MNE with the existing works, few existing network embedding models can be applied to address the BL-MNE directly. A new embedding learning framework is needed to learn the emerging network embedding vectors across multiple aligned networks synergistically.

To address all these challenges aforementioned, in this paper, we introduce a novel multiple aligned heterogeneous social network embedding framework, named "<u>D</u>eep al<u>I</u>gned autoencoder based e<u>Mb</u>E<u>d</u>ding" (DIME). To handle the heterogeneous link and attribute information in a unified analytic, we introduce the *aligned attribute augmented heterogeneous network* concept in this paper. From these heterogeneous networks a set of meta paths are introduced to represent the diverse connections among users in online social networks (via social links, other diverse connections, and various attributes). A set of *meta proximity* measures are defined for each of the meta paths denoting the closeness among users. The meta proximity information will be fed into a deep learning framework, which takes the input information from multiple aligned heterogeneous social networks simultaneously, to achieve the embedding feature vectors for all the users in these aligned networks. Based on the connection among users, framework DIME aims at embedding close user nodes to a close area in the low-dimensional feature space for each of the social networks respectively. Meanwhile, framework DIME also poses constraints on the feature vectors corresponding to the shared users across networks to map them to a relatively close region. In this way, information can be transferred from the mature networks to the emerging network and solve the *information sparsity* problem.

The remaining parts of this paper are organized as follows. We will provide the terminology definition and problem formulation in Section II. Information about the framework is available in Section III, which will be evaluated in Section IV. Finally, we will introduce the related works in Section V and conclude this paper in Section VI.

II. TERMINOLOGY DEFINITION AND PROBLEM FORMULATION

In this section, we will first introduce the definitions of several important terminologies, based on which we will then provide the formulation of the BL-MNE problem.

A. Terminology Definition

The social networks studied in this paper contain different categories of nodes and links, as well as very diverse attributes attached to the nodes. Formally, we can represent these network structured data as the *attributed heterogeneous social networks*.

Definition 1 (Attributed Heterogeneous Social Networks): The *attributed heterogeneous social network* can be represented as a graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{T})$, where $\mathcal{V} = \bigcup_i \mathcal{V}_i$ denotes the set of nodes belonging to various categories and $\mathcal{E} = \bigcup_i \mathcal{E}_i$ represents the set of diverse links among the nodes. What's more, $\mathcal{T} = \bigcup_i \mathcal{T}_i$ denotes the set of attributes attached to the nodes in $\mathcal{V}$. For user u in the network, we can represent the i_{th} type of attribute associated to u as $T_i(u)$, and all the attributes u has can be represented as $T(u) = \bigcup_i T_i(u)$.

The social network datasets used in this paper include Foursquare and Twitter. Formally, the Foursquare and Twitter can both be represented as the *attributed heterogeneous social networks* $G = (\mathcal{V}, \mathcal{E}, \mathcal{T})$, where $\mathcal{V} = \mathcal{U} \cup \mathcal{P}$ involves the user and post nodes, and $\mathcal{E} = \mathcal{E}_{u,u} \cup \mathcal{E}_{u,p}$ contains the links among users and those between users and posts. In addition, the nodes in $\mathcal{V}$ are also attached with a set of attributes, i.e., $\mathcal{T}$. For instance, for the posts written by users, we can obtain the contained textual contents, timestamps and checkins, which can all be represented as the attributes of the post nodes.

Between Foursquare and Twitter, there may exist a large number of shared common users, who can align the networks together. In this paper, we will follow the concept definitions proposed in [30], and call the user account correspondence relationships as the *anchor links*. Meanwhile, the networks connected by the *anchor links* are called the multiple *aligned attributed heterogeneous social networks* (or *aligned social networks* for short).

Definition 2 (Multiple Aligned Social Networks): Formally, given n attributed heterogeneous social networks $\{G^{(1)}, \cdots, G^{(n)}\}$ with shared users, they can be defined as *multiple aligned social networks* $\mathcal{G} = ((G^{(1)}, \cdots, G^{(n)}), (\mathcal{A}^{(1,2)}, \cdots, \mathcal{A}^{(n-1,n)}))$. Set $\mathcal{A}^{(i,j)}$ represents the anchor links between $G^{(i)}$ and $G^{(j)}$. User pair $(u^{(i)}, v^{(j)}) \in \mathcal{A}^{(i,j)}$ iff $u^{(i)}$ and $v^{(j)}$ are the accounts of the same user in networks $G^{(i)}$ and $G^{(j)}$ respectively.

For the Foursquare and Twitter social networks used in this paper, we can represent them as two aligned social networks $\mathcal{G} = ((G^{(1)}, G^{(2)}), (\mathcal{A}^{(1,2)}))$, which will be used as an example to illustrate the models. A simple extension of the proposed framework can be applied to *k aligned networks* very easily.

B. Problem Formulation

Problem Definition (BL-MNE Problem): Given two aligned networks $\mathcal{G} = ((G^{(1)}, G^{(2)}), (\mathcal{A}^{(1,2)}))$, where $G^{(1)}$ is an emerging network and $G^{(2)}$ is a mature network, BL-MNE aims at learning a mapping function $f^{(i)} : \mathcal{U}^{(i)} \rightarrow \mathbb{R}^{d^{(i)}}$ to project the user node in $G^{(i)}$ to a feature space of dimension $d^{(i)}$ ($d^{(i)} \ll |\mathcal{U}|^{(i)}$). The objective of mapping functions $f^{(i)}$ is to ensure the embedding results can preserve the network structural information, where similar user nodes will be projected to close regions. Furthermore, in the embedding process, BL-MNE also wants to transfer information between $G^{(2)}$ and $G^{(1)}$ to overcome the information sparsity problem in $G^{(1)}$.

III. PROPOSED METHOD

In this section, we will introduce the framework DIME in detail. At the beginning, we provide the notations used in the paper. After that, in Section III-B, we will talk about how to calculate the *meta proximity* scores among users based on information in the attributed heterogeneous social networks. With the *meta proximity* measures, the DIME framework will be introduced in Section III-C to obtain the embedding vectors of user nodes across aligned networks, where information from other aligned mature networks will be used to refine the embedding vectors in the emerging sparse network.

A. Notations

In the sequel, we will use the lower case letters (e.g., x) to represent scalars, lower case bold letters (e.g., $\mathbf{x}$) to denote column vectors, bold-face upper case letters (e.g., $\mathbf{X}$) to denote matrices, and upper case calligraphic letters (e.g., $\mathcal{X}$) to denote sets. Given a matrix $\mathbf{X}$, we denote $\mathbf{X}(i, :)$ and $\mathbf{X}(:, j)$ as the i_{th} row and j_{th} column of matrix $\mathbf{X}$ respectively. The (i_{th}, j_{th}) entry of matrix $\mathbf{X}$ can be denoted as either $X(i, j)$ or $X_{i,j}$, which will be used interchangeably in this paper. We use $\mathbf{X}^\top$ and $\mathbf{x}^\top$ to represent the transpose of

matrix $\mathbf{X}$ and vector $\mathbf{x}$. For vector $\mathbf{x}$, we represent its L_p-norm as $\|\mathbf{x}\|_p = (\sum_i |x_i|^p)^{\frac{1}{p}}$. The L_p-norm of matrix $\mathbf{X}$ can be represented as $\|\mathbf{X}\|_p = (\sum_{i,j} |X_{i,j}|^p)^{\frac{1}{p}}$. The element-wise product of vectors $\mathbf{x}$ and $\mathbf{y}$ of the same dimension is represented as $\mathbf{x} \odot \mathbf{y}$, while the element-wise product of matrices $\mathbf{X}$ and $\mathbf{Y}$ of the same dimensions is represented as $\mathbf{X} \odot \mathbf{Y}$.

B. Heterogeneous Network Meta Proximity

For each attributed heterogeneous social network, the closeness among users can be denoted by the friendship links among them, where friends tend to be closer compared with user pairs without connections. Meanwhile, for the users who are not directly connected by the friendship links, few existing embedding methods can figure out their closeness, as these methods are mostly built based on the direct friendship link only. In this section, we propose to infer the potential closeness scores among the users with the heterogeneous information in the networks based on meta path concept [16], which are formally called the *meta proximity* in the paper.

1) Friendship based Meta Proximity: In online social networks, the friendship links are the most obvious indicator of the social closeness among users. Online friends tend to be closer with each other compared with the user pairs who are not friends. Users' friendship links also carry important information about the local network structure information, which should be preserved in the embedding results. Based on such an intuition, we propose the *friendship based meta proximity* concept as follows.

Definition 3 (Friendship based Meta Proximity): For any two user nodes $u_i^{(1)}, u_j^{(1)}$ in an online social network (e.g., $G^{(1)}$), if $u_i^{(1)}$ and $u_j^{(1)}$ are friends in $G^{(1)}$, the *friendship based meta proximity* between $u_i^{(1)}$ and $u_j^{(1)}$ in the network is 1, otherwise the *friendship based meta proximity* score between them will be 0 instead. To be more specific, we can represent the *friendship based meta proximity* score between users $u_i^{(1)}, u_j^{(1)}$ as $p^{(1)}(u_i^{(1)}, u_j^{(1)}) \in \{0, 1\}$, where $p^{(1)}(u_i^{(1)}, u_j^{(1)}) = 1$ iff $(u_i^{(1)}, u_j^{(1)}) \in \mathcal{E}_{u,u}^{(1)}$.

Based on the above definition, the *friendship based meta proximity* scores among all the users in network $G^{(1)}$ can be represented as matrix $\mathbf{P}_{\Phi_0}^{(1)} \in \mathbb{R}^{|\mathcal{U}^{(1)}| \times |\mathcal{U}^{(1)}|}$, where entry $P_{\Phi_0}^{(1)}(i, j)$ equals to $p^{(1)}(u_i^{(1)}, u_j^{(1)})$. Here Φ_0 denotes the simplest meta path of length 1 in the form $U \xrightarrow{\text{follow}} U$, and its formal definition will be introduced in the following subsection.

When network $G^{(1)}$ is an emerging online social network which has just started to provide services for a very short time, the friendship links among users in $G^{(1)}$ tend to be very limited (majority of the users are isolated in the network with few social connections). In other words, the *friendship based meta proximity* matrix $\mathbf{P}_{\Phi_0}^{(1)}$ will be extremely sparse, where few entries will have value 1 and most of the entries are 0s. With such a sparse matrix, most existing embedding models will fail to work. The reason is that the sparse friendship information available in the network can hardly categorize the relative closeness relationships among the users (especially for those who are even not connected by friendship links), which renders these existing embedding models may project all the nodes to random regions.

To overcome such a problem, besides the social links, we propose to calculate the proximity scores for the users with the diverse link and attribute information in the heterogeneous networks in this paper. Based on a new concept named *attribute augmented meta path*, a set of *meta proximity* measures will be defined with each of the meta paths, which will be introduced in the following sections.

ID	Notation	Heterogeneous Network Meta Path	Semantics
Φ_0	U → U	User $\xrightarrow{follow}$ User	Follow
Φ_1	U → U → U	User $\xrightarrow{follow}$ User $\xrightarrow{follow}$ User	Follower of Follower
Φ_2	U → U ← U	User $\xrightarrow{follow}$ User $\xrightarrow{follow^{-1}}$ User	Common Out Neighbor
Φ_3	U ← U → U	User $\xrightarrow{follow^{-1}}$ User $\xrightarrow{follow}$ User	Common In Neighbor
Φ_4	U ← U ← U	User $\xrightarrow{follow^{-1}}$ User $\xrightarrow{follow^{-1}}$ User	Followee of Followee
Φ_5	U → P → W ← P ← U	User $\xrightarrow{write}$ Post $\xrightarrow{have}$ Word $\xrightarrow{have^{-1}}$ Post $\xrightarrow{write^{-1}}$ User	Posts Containing Common Words
Φ_6	U → P → T ← P ← U	User $\xrightarrow{write}$ Post $\xrightarrow{have}$ Time $\xrightarrow{have^{-1}}$ Post $\xrightarrow{write^{-1}}$ User	Posts Containing Common Timestamps
Φ_7	U → P → L ← P ← U	User $\xrightarrow{write}$ Post $\xrightarrow{have}$ Location $\xrightarrow{have^{-1}}$ Post $\xrightarrow{write^{-1}}$ User	Posts Attaching Common Location Check-ins

2) Attribute Augmented Meta Path: To handle the diverse links and attributes simultaneously in a unified analytic, we propose to treat the attributes as nodes as well and introduce the *attribute augmented network*. If a node has certain attributes, a new type of link "*have*" will be added to connected the node and the newly added attribute node. The structure of the *attribute augmented network* can be described with the *attribute augmented network schema* as follows.

Definition 4 (Attribute Augmented Network Schema): Formally, the network schema of a given online social network $G^{(1)} = (\mathcal{V}, \mathcal{E})$ can be represented as $S_{G^{(1)}} = (\mathcal{N}_\mathcal{V} \cup \mathcal{N}_\mathcal{T}, \mathcal{R}_\mathcal{E} \cup \{have\})$, where $\mathcal{N}_\mathcal{V}$ and $\mathcal{N}_\mathcal{T}$ denote the set of node and attribute categories in the network, while $\mathcal{R}_\mathcal{E}$ represents the set of link types in the network, and $\{have\}$ represents the relationship between node and attribute node types.

For instance, about the *attributed heterogeneous social network* introduced after Definition 1 in Section II, we can represent its network schema as $S_{G^{(1)}} = (\mathcal{N}_\mathcal{V} \cup \mathcal{N}_\mathcal{T}, \mathcal{R}_\mathcal{E} \cup \{have\})$. The node type set $\mathcal{N}_\mathcal{V}$ involves node types $\{User, Post\}$ (or $\{U, P\}$ for simplicity), while the attribute type set $\mathcal{N}_\mathcal{T}$ includes $\{Word, Time, Location\}$ (or $\{W, T, L\}$ for short). As to the link types involved in the network, the link type set $\mathcal{R}_\mathcal{E}$ contains $\{follow, write\}$, which represents the friendship link type and the write link type respectively.

Based on the *attribute augmented network schema*, we can represent the general correlation among users (especially those who are directly connected by friendship links) with the *attributed augmented meta path* starting and ending with the user node type.

Definition 5 (Attribute Augmented Meta Path): Given a network schema $S_{G^{(1)}}$, the *attribute augmented meta path* denotes a sequence of node/attribute types connected by the link types or the "have" relation type (between node and attribute type). Formally, the *attribute augmented meta path* (of length $k-1$, $k \geq 2$) can be represented as $\Phi : N_1 \xrightarrow{R_1} N_2 \xrightarrow{R_2} \cdots \xrightarrow{R_{k-1}} N_k$, where $N_1, \cdots, N_k \in \mathcal{N}_\mathcal{V} \cup \mathcal{N}_\mathcal{T}$ and $R_1, \cdots, R_{k-1} \in \mathcal{R}_\mathcal{E} \cup \mathcal{R}_\mathcal{E}^{-1} \cup \{have, have^{-1}\}$ (superscript -1 denotes the reverse of relation type direction). In the case that $N_1 = N_k = U$, i.e., meta paths starts and ends with the user node type, the meta paths will be called the *social meta paths* specifically.

Based on the above definition, a set of different *social meta path* $\{\Phi_0, \Phi_1, \Phi_2, \cdots, \Phi_7\}$ can be extracted from the network, whose notations, concrete representations and the physical meanings are illustrated in Table II. Here, meta paths $\Phi_0 - \Phi_4$ are all based on the user node type and follow link type; meta paths $\Phi_5 - \Phi_7$ involve the user, post node type, attribute node type, as well as the *write* and *have* link type. Based on each of the meta paths, there will exist a set of concrete meta path instances connecting users in the networks. For instance, given a user pair u and v, they may have been checked-in at 5 different common locations, which will introduce 5 concrete meta path instance of meta path Φ_7 connecting u and v indicating their strong closeness (in location check-ins). In the next subsection, we will introduce how to calculate the proximity score for the users based on these extracted meta paths.

3) Heterogeneous Network Meta Proximity: The set of *attribute augmented social meta paths* $\{\Phi_0, \Phi_1, \Phi_2, \cdots, \Phi_7\}$ extracted in the previous subsection create different kinds of correlations among users (especially for those who are not directed connected by friendship links). With these *social meta paths*, different types of proximity scores among the users can be captured. For instance, for the users who are not friends but share lots of common friends, they may also know each other and can be close to each other; for the users who frequently checked-in at the same places, they tend to be more close to each other compared with those isolated ones with nothing in common. Therefore, these meta paths can help capture much broader network structure information compared with the local structure captured by the *friendship based meta proximity* covered in Section III-B1. In this part, we will introduce the method to calculate the proximity scores among users based on these *social meta paths*.

As shown in Table II, all the social meta paths extracted from the networks can be represented as set $\{\Phi_0, \Phi_1, \cdots, \Phi_7\}$. Given a pair of users, e.g., $u_i^{(1)}$ and $u_j^{(1)}$, based on meta path $\Phi_k \in \{\Phi_0, \Phi_1, \cdots, \Phi_7\}$, we can represent the set of meta path instances connecting $u_i^{(1)}$ and $u_j^{(1)}$ as $\mathcal{P}_{\Phi_k}^{(1)}(u_i^{(1)}, u_j^{(1)})$. Users $u_i^{(1)}$ and $u_j^{(1)}$ can have multiple meta path instances going into/out from them. Formally, we can represent all the meta path instances going out from user $u_i^{(1)}$ (or going into $u_j^{(1)}$), based on meta path Φ_k, as set $\mathcal{P}_{\Phi_k}^{(1)}(u_i^{(1)}, \cdot)$ (or $\mathcal{P}_{\Phi_k}^{(1)}(\cdot, u_j^{(1)})$). The proximity score between $u_i^{(1)}$ and $u_j^{(1)}$ based on meta path Φ_k can be represented as the following *meta proximity* concept formally.

Definition 6 (Meta Proximity): Based on meta path Φ_k, the meta proximity between users $u_i^{(1)}$ and $u_j^{(1)}$ in $G^{(1)}$ can be represented as

$$p_{\Phi_k}^{(1)}(u_i^{(1)}, u_j^{(1)}) = \frac{2|\mathcal{P}_{\Phi_k}^{(1)}(u_i^{(1)}, u_j^{(1)})|}{|\mathcal{P}_{\Phi_k}^{(1)}(u_i^{(1)}, \cdot)| + |\mathcal{P}_{\Phi_k}^{(1)}(\cdot, u_j^{(1)})|}.$$

Meta proximity considers not only the meta path instances between users but also penalizes the number of meta path instances going out from/into $u_i^{(1)}$ and $u_j^{(1)}$ at the same time. It is also reasonable. For instance, sharing some common location check-ins with some extremely active users (who have thousands of checkins) may not necessarily indicate closeness with them, since they may have common check-ins with almost all other users simply due to his very large check-in record volume instead of their closeness.

With the above meta proximity definition, we can represent the meta proximity scores among all users in the network $G^{(1)}$ based on meta path Φ_k as matrix $\mathbf{P}_{\Phi_k}^{(1)} \in \mathbb{R}^{|\mathcal{U}^{(1)}| \times |\mathcal{U}^{(1)}|}$, where entry $P_{\Phi_k}^{(1)}(i, j) = p_{\Phi_k}^{(1)}(u_i^{(1)}, u_j^{(1)})$. All the meta proximity matrices defined for network $G^{(1)}$ can be represented as $\{\mathbf{P}_{\Phi_k}^{(1)}\}_{\Phi_k}$. Based on the meta paths extracted for network $G^{(2)}$, similar matrices can be defined as well, which can be denoted as $\{\mathbf{P}_{\Phi_k}^{(2)}\}_{\Phi_k}$.
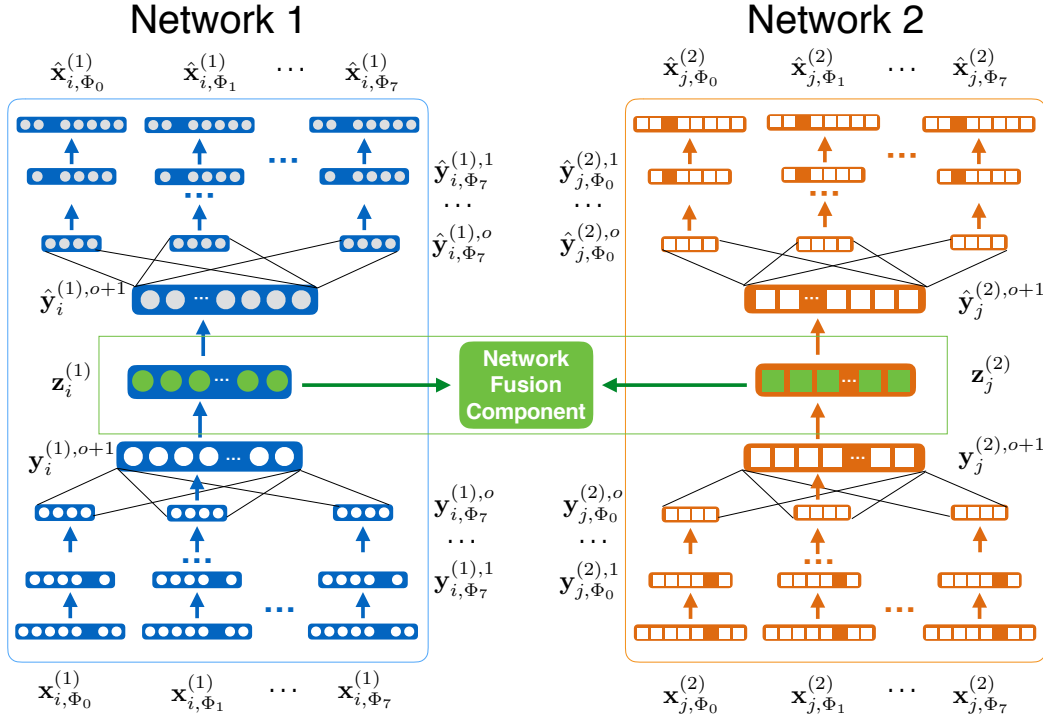
Fig. 1. The DIME Framework.

C. Deep Network Synergistic Embedding

With these calculated *meta proximity* introduced in the previous section, we will introduce the embedding framework DIME in this part. DIME is based on the *aligned auto-encoder model*, which extends the traditional *deep auto-encoder model* to the *multiple aligned heterogeneous networks* scenario. To make this paper self-contained, we will first briefly introduce some background knowledge about the auto-encoder model first in Section III-C1. After that, we will talk about the embedding model component for one single heterogeneous network in Section III-C2, which takes the various meta proximity matrices as input. DIME effectively couples the embedding process of the emerging network with other aligned mature networks, where cross-network information exchange and refinement is achieved via the loss term defined based on the anchor links.

1) Deep Auto-Encoder Model Review: Auto-encoder is an unsupervised neural network model, which projects the instances (in original feature representations) into a lower-dimensional feature space via a series of non-linear mappings. Auto-encoder model involves two steps: encoder and decoder. The encoder part projects the original feature vectors to the objective feature space, while the decoder step recovers the latent feature representation to a reconstruction space. In auto-encoder model, we generally need to ensure that the original feature representation of instances should be as similar to the reconstructed feature representation as possible.

Formally, let $\mathbf{x}_i$ represent the original feature representation of instance i, and $\mathbf{y}_i^1, \mathbf{y}_i^2, \cdots, \mathbf{y}_i^o$ be the latent feature representation of the instance at hidden layers $1, 2, \cdots, o$ in the encoder step, the encoding result in the objective feature space can be represented as $\mathbf{z}_i \in \mathbb{R}^d$ with dimension d. Formally, the relationship between these variables can be represented with the following equations:

$$\begin{cases} \mathbf{y}_i^1 & = \sigma(\mathbf{W}^1 \mathbf{x}_i + \mathbf{b}^1), \\ \mathbf{y}_i^k & = \sigma(\mathbf{W}^k \mathbf{y}_i^{k-1} + \mathbf{b}^k), \forall k \in \{2, 3, \cdots, o\}, \\ \mathbf{z}_i & = \sigma(\mathbf{W}^{o+1} \mathbf{y}_i^o + \mathbf{b}^{o+1}). \end{cases}$$

Meanwhile, in the decoder step, the input will be the latent feature vector $\mathbf{z}_i$ (i.e., the output of the encoder step), and the final output will be the reconstructed vector $\hat{\mathbf{x}}_i$. The latent feature vectors at

each hidden layers can be represented as $\hat{\mathbf{y}}_i^o, \hat{\mathbf{y}}_i^{o-1}, \cdots, \hat{\mathbf{y}}_i^1$. The relationship between these vector variables can be denoted as

$$\begin{cases} \hat{\mathbf{y}}_i^o & = \sigma(\hat{\mathbf{W}}^{o+1} \mathbf{z}_i + \hat{\mathbf{b}}^{o+1}), \\ \hat{\mathbf{y}}_i^{k-1} & = \sigma(\hat{\mathbf{W}}^k \hat{\mathbf{y}}_i^k + \hat{\mathbf{b}}^k), \forall k \in \{2, 3, \cdots, o\}, \\ \hat{\mathbf{x}}_i & = \sigma(\hat{\mathbf{W}}^1 \hat{\mathbf{y}}_i^1 + \hat{\mathbf{b}}^1). \end{cases}$$

The objective of the auto-encoder model is to minimize the loss between the original feature vector $\mathbf{x}_i$ and the reconstructed feature vector $\hat{\mathbf{x}}_i$ of all the instances in the network. Formally, the loss term can be represented as

$$\mathcal{L} = \sum_i \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2.$$

2) Deep DIME-SH Model: When applying the auto-encoder model for one single homogeneous network, e.g., for $G^{(1)}$, we can fit the model with the node meta proximity feature vectors, i.e., rows corresponding to users in matrix $\mathbf{P}_{\Phi_0}^{(1)}$ (introduced in Section III-B1). In the case that $G^{(1)}$ is heterogeneous, multiple node *meta proximity* matrices have been defined before (i.e., $\{\mathbf{P}_{\Phi_0}^{(1)}, \mathbf{P}_{\Phi_1}^{(1)}, \cdots, \mathbf{P}_{\Phi_7}^{(1)}\}$), how to fit these matrices simultaneously to the auto-encoder models is an open problem. In this part, we will introduce the single-heterogeneous-network version of framework DIME, namely DIME-SH, which will be used as an important component of framework DIME as well. For each user node in the network, DIME-SH computes the embedding vector based on each of the proximity matrix independently first, which will be further fused to compute the final latent feature vector in the output hidden layer.

As shown in the architecture in Figure 1 (either the left component for network 1 or the right component for network 2), about the same instance, DIME-SH 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 $\mathbf{z}_i^{(1)} \in \mathbb{R}^{d^{(1)}}$ for user $u_i^{(1)} \in \mathcal{V}^{(1)}$. 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 $\mathbf{x}_{i,\Phi_k}^{(1)}$, which denotes the row corresponding to users $u_i^{(1)}$ in matrix $\mathbf{P}_{\Phi_k}^{(1)}$

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 $\{\mathbf{y}_{i,\Phi_k}^{(1),1}, \mathbf{y}_{i,\Phi_k}^{(1),2}, \cdots, \mathbf{y}_{i,\Phi_k}^{(1),o}\}$.

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

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

In the fusion and dispatch steps, full connection layers are used in order to incorporate all the information captured by all the meta paths.

What's more, since the input feature vectors are extremely sparse (lots of the entries are 0s), simply feeding them to the model may lead to some trivial solutions, like $\mathbf{0}$ vectors for both $\mathbf{z}_i^{(1)}$ and the decoded vectors $\hat{\mathbf{x}}_{i,\Phi_k}^{(1)}$. 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

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

where vector $\mathbf{b}_{i,\Phi_k}^{(1)}$ is the weight vector corresponding to feature vector $\mathbf{x}_{i,\Phi_k}^{(1)}$. Entries in vector $\mathbf{b}_{i,\Phi_k}^{(1)}$ are filled with value 1s except the entries corresponding to non-zero element in $\mathbf{x}_{i,\Phi_k}^{(1)}$, which will be assigned with value γ ($\gamma > 1$ denoting a larger weight to fit these features). Here, we need to add a remark that "simply discarding the entries corresponding zero values in the input vectors from the loss function" will not work here, since it will allow the model to decode there entries to any random values, which will not be what we want. In a similar way, we can define the loss function for the embedding result in network $G^{(2)}$, which can be formally represented as $\mathcal{L}^{(2)}$.

3) Deep DIME Framework: DIME-SH has incorporate all these heterogeneous information in the model building, the meta proximity calculated based on which can help differentiate the closeness among different users. However, for the emerging networks which just start to provide services, the information sparsity problem may affect the performance of DIME-SH significantly. In this part, we will introduce DIME, which couples the embedding process of the emerging network with another mature aligned network. By accommodating the embedding between the aligned networks, information can be transferred from the aligned mature network to refine the embedding results in the emerging network effectively. The complete architecture of DIME is shown in Figure 1, 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), we can represent the embedding results 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 $\mathbf{Z}^{(1)}$

TABLE III
PROPERTIES OF THE HETEROGENEOUS NETWORKS

| | property | network | |
		Twitter	**Foursquare**
# node	user	5,223	5,392
	tweet/tip	9,490,707	48,756
	location	297,182	38,921
# link	friend/follow	164,920	76,972
	write	9,490,707	48,756
	locate	615,515	48,756

(or the j_{th} row of matrix $\mathbf{Z}^{(2)}$) denotes the encoded feature vector of user $u_i^{(1)}$ in $G^{(1)}$ (or $u_j^{(2)}$ in $G^{(2)}$). If $u_i^{(1)}$ and $u_j^{(2)}$ are the same user, i.e., $(u_i^{(1)}, u_j^{(2)}) \in \mathcal{A}^{(1,2)}$, by placing vectors $\mathbf{Z}^{(1)}(i,:)$ and $\mathbf{Z}^{(2)}(j,:)$ in a close region in the embedding space, we can use the information from $G^{(2)}$ 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, we introduce 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., $(u_i^{(1)}, u_j^{(2)}) \in \mathcal{A}^{(1,2)}$. 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

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

By minimizing the *information fusion loss* function $\mathcal{L}^{(1,2)}$, we can use the anchor users' embedding vectors from the mature network $G^{(2)}$ to adjust his embedding vectors in the emerging network $G^{(1)}$. Even through in such a process the embedding vector in $G^{(2)}$ can be undermined by $G^{(1)}$, it will not be a problem since $G^{(1)}$ is our target network and we only care about the embedding result of the emerging network $G^{(1)}$ in the paper.

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*, which can be denoted as

$$
\mathcal{L}(G^{(1)}, G^{(2)}) = \mathcal{L}^{(1)} + \mathcal{L}^{(2)} + \alpha \cdot \mathcal{L}^{(1,2)} + \beta \cdot \mathcal{L}_{reg}.
$$

Parameters α and β denote the weights of the *information fusion loss* term and the regularization term. In the objective function, term $\mathcal{L}_{reg}$ is added to the above objective function to avoid overfitting, which can be formally represented as

$$
\begin{cases}
\mathcal{L}_{reg} = \mathcal{L}_{reg}^{(1)} + \mathcal{L}_{reg}^{(2)} + \mathcal{L}_{reg}^{(1,2)}, \\
\mathcal{L}_{reg}^{(1)} = \sum_i^{o^{(1)}+2} \sum_{\Phi_k \in \{\Phi_0, \cdots, \Phi_7\}} \left(\left\| \mathbf{W}_{\Phi_k}^{(1),i} \right\|_F^2 + \left\| \hat{\mathbf{W}}_{\Phi_k}^{(1),i} \right\|_F^2 \right), \\
\mathcal{L}_{reg}^{(2)} = \sum_i^{o^{(2)}+2} \sum_{\Phi_k \in \{\Phi_0, \cdots, \Phi_7\}} \left(\left\| \mathbf{W}_{\Phi_k}^{(2),i} \right\|_F^2 + \left\| \hat{\mathbf{W}}_{\Phi_k}^{(2),i} \right\|_F^2 \right), \\
\mathcal{L}_{reg}^{(1,2)} = \left\| \mathbf{W}^{(1,2)} \right\|_F^2.
\end{cases}
$$

To optimize the above objective function, we utilize Stochastic Gradient Descent (SGD). To be more specific, the training process involves multiple epochs. In each epoch, the training data is shuffled and a minibatch of the instances are sampled to update the parameters with SGD. Such a process continues until either convergence or the training epochs have been finished.

TABLE IV
LINK PREDICTION RESULT OF THE COMPARISON METHODS (PARAMETER λ CHANGES IN $\{10\%, 20\%, \cdots, 100\%\}$, $\theta = 1$).

metric	method	Sampling Ratio λ									
		10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
AUC	DIME	**0.792±0.007**	**0.822±0.006**	**0.838±0.005**	**0.843±0.003**	**0.847±0.003**	**0.850±0.003**	**0.852±0.002**	**0.850±0.004**	**0.852±0.003**	**0.852±0.004**
	DIME-SH	0.774±0.006	0.795±0.005	0.802±0.006	0.809±0.004	0.815±0.005	0.822±0.005	0.827±0.006	0.826±0.005	0.830±0.004	0.833±0.003
	Auto-encoder	0.697±0.006	0.731±0.006	0.752±0.005	0.761±0.005	0.761±0.005	0.763±0.004	0.763±0.004	0.770±0.003	0.773±0.005	0.777±0.005
	LINE	0.694±0.008	0.716±0.010	0.731±0.007	0.738±0.005	0.741±0.006	0.744±0.005	0.748±0.005	0.748±0.008	0.750±.003	0.750±0.006
	DeepWalk	0.671±0.010	0.661±0.011	0.670±0.009	0.675±0.009	0.682±0.005	0.687±0.004	0.701±0.007	0.718±0.007	0.733±0.008	0.747±0.006
Accuracy	DIME	**0.719±0.006**	**0.748±0.005**	**0.763±0.004**	**0.767±0.003**	**0.773±0.003**	**0.775±0.004**	**0.777±0.003**	**0.775±0.003**	**0.777±0.004**	**0.777±0.004**
	DIME-SH	0.704±0.007	0.723±0.004	0.728±0.006	0.737±0.003	0.739±0.006	0.747±0.005	0.753±0.006	0.754±0.006	0.757±0.005	0.761±0.003
	Auto-encoder	0.642±0.005	0.668±0.005	0.684±0.005	0.692±0.005	0.691±0.005	0.691±0.004	0.691±0.004	0.699±0.004	0.700±0.005	0.703±0.005
	LINE	0.637±0.005	0.666±0.004	0.676±0.008	0.676±0.005	0.677±0.004	0.679±0.006	0.679±0.005	0.679±0.008	0.681±0.003	0.682±0.007
	DeepWalk	0.632±0.008	0.626±0.009	0.633±0.008	0.634±0.008	0.637±0.006	0.641±0.004	0.655±0.007	0.669±0.006	0.680±0.006	0.687±0.004
Recall	DIME	0.641±0.008	0.702±0.011	0.732±0.007	0.746±0.008	**0.755±0.008**	**0.761±0.006**	**0.767±0.006**	**0.768±0.003**	**0.771±0.005**	**0.772±0.008**
	DIME-SH	0.641±0.011	0.689±0.006	0.692±0.010	0.703±0.009	0.707±0.009	0.713±0.013	0.720±0.010	0.719±0.012	0.725±0.009	0.731±0.008
	Auto-encoder	0.564±0.016	0.649±0.016	0.714±0.017	0.743±0.013	0.726±0.012	0.680±0.009	0.671±0.007	0.681±0.005	0.680±0.008	0.687±0.007
	LINE	**0.819±0.008**	**0.770±0.007**	**0.800±0.008**	**0.749±0.011**	0.740±0.008	0.731±0.010	0.724±0.007	0.721±0.007	0.715±0.005	0.716±0.009
	DeepWalk	0.645±0.020	0.658±0.020	0.678±0.016	0.681±0.016	0.680±0.016	0.682±0.010	0.692±0.009	0.702±0.008	0.706±0.005	0.707±0.007
F1	DIME	**0.700±0.007**	**0.735±0.008**	**0.756±0.006**	**0.762±0.006**	**0.769±0.006**	**0.772±0.005**	**0.775±0.004**	**0.774±0.004**	**0.776±0.005**	**0.776±0.006**
	DIME-SH	0.684±0.009	0.711±0.005	0.718±0.006	0.728±0.006	0.731±0.007	0.738±0.008	0.744±0.007	0.745±0.009	0.749±0.006	0.753±0.005
	Auto-encoder	0.612±0.009	0.662±0.009	0.693±0.007	0.707±0.007	0.701±0.007	0.688±0.006	0.685±0.006	0.694±0.006	0.694±0.007	0.698±0.008
	LINE	0.693±0.006	0.698±0.005	0.712±0.008	0.698±0.007	0.696±0.006	0.695±0.009	0.693±0.006	0.692±0.009	0.692±0.005	0.693±0.009
	DeepWalk	0.636±0.011	0.637±0.014	0.648±0.010	0.650±0.010	0.652±0.010	0.655±0.006	0.667±0.008	0.679±0.006	0.688±0.006	0.693±0.005

IV. EXPERIMENTS

To demonstrate the effectiveness of the learnt embedding feature vectors, extensive experiments have been done on real-world aligned heterogeneous social networks, Foursquare and Twitter. Two different tasks are done in this section for embedding result evaluation purposes, which include *link prediction* and *community detection*. In this section, we will provide some basic descriptions about the aligned heterogeneous social network dataset first. After that, we will introduce the experimental settings, covering the comparison embedding methods, as well as the experimental settings, and evaluation metrics for *link prediction* and *community detection* tasks. Finally, we will show the experimental results about *link prediction* and *community detection*, followed by the parameter analysis.

A. Dataset Description

The data used in the experiments include two *aligned heterogeneous social networks* Foursquare and Twitter simultaneously. The basic statistical information about the Foursquare and Twitter datasets is available in Table III. The data crawling strategy and method is introduced in great deital in [8], [30].

- *Twitter*: Twitter is a famous *micro-blogging site* that allows users to write, read and share posts with their friends online. We have crawled $5,223$ Twitter users, and $164,920$ follow links among them. These crawled Twitter users have posted $9,490,707$ tweets, among which $615,515$ have location checkins.
- *Foursquare*: Foursquare is a famous *location based social network* (LBSN), which provides users with various kinds of location-related services. From Foursquare, we have crawled $5,392$ users together with $76,972$ friendship links among them. These Foursquare users have written $48,756$ posts which all attach location checkins. Among these $5,392$ crawled Foursquare users, $3,388$ of them are aligned by anchor links with Twitter.

In the experiments, we will use Foursquare as the emerging network and Twitter as the aligned mature network, since Twitter has more dense information than Foursquare. The results for the reverse case (Foursquare: mature; Twitter: emerging) are not shown here due to the limited space.

Source Code: The source code of DIME is available at site: http://www.ifmlab.org/files/code/Aligned-Autoencoder.zip.

B. Experimental Settings

In this paper, we are mainly focused on studying the embedding models, and different network embedding comparison methods will be introduced first. After that, we will introduce the experimental settings for both *link prediction* and *community detection* tasks, which will be used as the evaluation tasks to determine whether the embedding results are good or not. A set of frequently used evaluation metrics for link prediciton and community detection will be introduced afterwards.

1) Embedding Comparison Methods: The network embedding models compared in the experiments are listed as follows

- DIME: DIME is the synergistic embedding model for multiple aligned heterogeneous networks introduced in this paper. DIME preserves both the local and global network structure with a set of *meta proximity* calculated from each of the heterogeneous network. DIME transfers the information from the aligned mature networks to the emerging network with the anchor links, which accommodate the learnt embedding feature vectors for the anchor users in the aligned networks.
- DIME-SH: DIME-SH is a variant model of DIME proposed in this paper, which preserves both the local and global network structure with a set of *meta proximity* based on the heterogeneous networks. DIME-SH effectively fuses the heterogeneous information inside the network, where the fusion weight of information in different categories can be learnt automatically.
- *Auto-encoder Model*: The AUTO-ENCODER model proposed in [2] can project the instances into a low-dimensional feature space. In the experiments, we build the AUTO-ENCODER model merely based on the friendship link among users, where the feature vector for each user is his/her social adjacency vector. Here, we also adjust the loss term for AUTO-ENCODER by weighting the non-zero features more with parameter γ as introduced in Section III-C2.
- *LINE Model*: The LINE model is a scalable network embedding model proposed in [18], which optimizes an objective function that preserves both the local and global network structures. LINE also uses a edge-sampling algorithm to addresses the limitation of the classical stochastic gradient descent, which improves the inference effectiveness and the efficiency greatly.
- *DeepWalk Model*: The DEEPWALK model [14] extends the word2vec model [12] to the network embedding scenario. DEEPWALK uses local information obtained from truncated random walks to learn latent representations.

2) Link Prediction Experimental Setting: Given the emerging network, from which we can obtain all the existing links inside the network as the positive set. Meanwhile, from the network, a subset of the non-existing links are randomly sampled as the negative set

TABLE V

COMMUNITY DETECTION RESULT OF THE COMPARISON METHODS (PARAMETER λ CHANGES IN $\{10\%, 20\%, \cdots, 100\%\}$, $k = 10$).

metric	method	Sampling Ratio λ									
		10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Density	DIME	**0.002±0.000**	**0.003±0.000**	**0.003±0.000**	**0.003±0.000**	**0.004±0.000**	**0.003±0.000**	**0.004±0.000**	**0.003±0.000**	**0.003±0.000**	**0.004±0.000**
	DIME-SH	**0.002±0.000**	0.002±0.000	**0.003±0.000**	**0.003±0.000**	0.003±0.000	**0.003±0.000**	0.003±0.000	**0.003±0.000**	**0.003±0.000**	0.003±0.000
	Auto-encoder	**0.002±0.000**	0.002±0.000	0.002±0.000	0.002±0.000	0.003±0.000	**0.003±0.000**	0.003±0.000	**0.003±0.000**	**0.003±0.000**	0.003±0.000
	LINE	0.001±0.000	0.002±0.000	0.002±0.000	0.002±0.000	0.002±0.000	0.002±0.000	0.002±0.000	0.002±0.000	0.002±0.000	0.002±0.000
	DeepWalk	0.001±0.000	0.001±0.000	0.001±0.000	0.001±0.000	0.001±0.000	0.001±0.000	0.001±0.000	0.001±0.000	0.001±0.000	0.001±0.000
Separability	DIME	0.230±0.007	**0.296±0.027**	**0.360±0.021**	**0.369±0.028**	**0.440±0.023**	**0.381±0.028**	**0.466±0.022**	0.347±0.004	**0.398±0.025**	**0.414±0.018**
	DIME-SH	**0.251±0.033**	0.235±0.007	0.288±0.013	0.294±0.010	0.311±0.020	0.282±0.018	0.298±0.007	**0.370±0.011**	0.362±0.010	0.372±0.017
	Auto-encoder	0.247±0.031	0.181±0.012	0.201±0.016	0.251±0.020	0.259±0.013	0.272±0.014	0.272±0.019	0.296±0.012	0.281±0.019	0.271±0.011
	LINE	0.132±0.003	0.153±0.010	0.167±0.004	0.157±0.007	0.179±0.008	0.186±0.007	0.203±0.007	0.200±0.011	0.190±0.010	0.221±0.013
	DeepWalk	0.113±0.001	0.117±0.002	0.120±0.002	0.121±0.003	0.123±0.002	0.121±0.001	0.123±0.002	0.124±0.003	0.124±0.002	0.123±0.002
Coverage	DIME	0.187±0.005	**0.228±0.016**	**0.264±0.011**	**0.269±0.015**	**0.306±0.011**	**0.276±0.014**	**0.318±0.011**	0.258±0.002	**0.285±0.013**	**0.292±0.009**
	DIME-SH	**0.200±0.021**	0.190±0.005	0.224±0.008	0.227±0.006	0.237±0.011	0.220±0.011	0.229±0.004	**0.270±0.006**	0.266±0.005	0.271±0.009
	Auto-encoder	0.198±0.020	0.153±0.009	0.167±0.011	0.201±0.013	0.206±0.008	0.214±0.009	0.213±0.012	0.228±0.007	0.219±0.012	0.213±0.007
	LINE	0.117±0.003	0.133±0.008	0.143±0.003	0.136±0.005	0.152±0.006	0.157±0.005	0.168±0.005	0.167±0.008	0.160±0.007	0.181±0.009
	DeepWalk	0.102±0.001	0.105±0.001	0.107±0.002	0.108±0.002	0.110±0.001	0.108±0.001	0.110±0.001	0.110±0.002	0.111±0.002	0.110±0.001
Expansion	DIME	0.813±0.005	**0.772±0.016**	**0.736±0.011**	**0.731±0.015**	0.694±0.011	**0.724±0.014**	0.682±0.011	0.742±0.002	**0.715±0.013**	**0.708±0.009**
	DIME-SH	**0.800±0.021**	0.810±0.005	0.776±0.008	0.773±0.006	0.763±0.011	0.780±0.011	0.771±0.004	**0.730±0.006**	0.734±0.005	0.729±0.009
	Auto-encoder	0.802±0.020	0.847±0.009	0.833±0.011	0.799±0.013	0.794±0.008	0.786±0.009	0.787±0.012	0.772±0.007	0.781±0.012	0.787±0.007
	LINE	0.883±0.003	0.867±0.008	0.857±0.003	0.864±0.005	0.848±0.006	0.843±0.005	0.832±0.005	0.833±0.008	0.840±0.007	0.819±0.009
	DeepWalk	0.898±0.001	0.895±0.001	0.893±0.002	0.892±0.002	0.890±0.001	0.892±0.001	0.890±0.001	0.890±0.002	0.889±0.002	0.890±0.001

according to the *negative positive sampling ratio* $\theta \in \{1, 2, \cdots, 10\}$. Here $\theta = 1$ denotes that the negative set is of the same size as the positive set. Meanwhile, $\theta = 10$ represents the negative set is 9 times larger than the positive set. The positive and negative sets are divided into two subsets with 10-fold cross validation, where 9 folds are used as the training set and 1 fold is used as the testing set.

To denote different degrees of information sparsity, the emerging network is further sampled to remove information randomly from the network, which is controlled by the *sampling ratio* $\lambda \in \{10\%, 20\%, \cdots, 100\%\}$. Here, the sampling denotes removing the positive links in the training set, as well as the posts from the emerging network to make the network sparse. $\lambda = 10\%$ denotes 10% of the information is preserved (90% of the information is randomly removed); while $\lambda = 100\%$ denotes that all the information is preserved in the emerging network. The network embedding is learnt based on the training set and network after sampling.

With the learnt embedding feature vectors, the remaining links in the training set is used to build a supervised link prediction model. For each link (u, v) in the training set, the embedding feature vector of the nodes u and v are concatenated as the link feature vector. Depending on whether link (u, v) appears in the positive set or negative set, (u, v) will be assigned with the $+1$ or -1 label. SVM is used as the base classifier for all the embedding models. We train SVM with the training set, and then apply the trained SVM to the testing set to infer the labels and the formation probabilities of these links. By comparing the prediction labels (and inference probabilities) with the ground truth labels, we can evaluate the performance of the embedding models with different kinds of evaluation metrics to be introduced in the next subsection.

In the experiments, 7 hidden layers are involved in framework DIME (3 hidden layers in encoder step, 3 in decoder step, and 1 fusion hidden layer). The number neuron in these hidden layers are 500, 50, 50×7, 50, 50×7, 50 and 500 respectively. Epoch is 600 and the batch size is 64. The parameters $\alpha = 1.0$, $\beta = 0.02$ and $\gamma = 100.0$ are used in the experiments.

3) Link Prediction Evaluation Metrics: By comparing the link prediction results in the testing set, i.e., the inference probabilities, with the ground truth labels, the performance of different embedding models can be evaluated by AUC as the metric. Meanwhile, based on the prediction labels, we can evaluate the performance of these embedding models with Recall, F1 and Accuracy as the metrics.

4) Community Detection Experimental Setting: Different from *link prediction*, *community detection* is an unsupervised learning task, where no training set is needed. Based on the whole network, we randomly sample a subset of information, i.e., follow links and posts, from the network controlled by the *sampling ratio* $\lambda \in \{10\%, 20\%, \cdots, 100\%\}$. Based on the sampled network, we learn the embedding of the emerging network and get the embedded feature vector for each user in the emerging network. KMeans is applied as the base clustering model to partition the users into different clusters based on their learnt embedding feature vectors. We evaluate the performance of the embedding methods by comparing the clustering results with the original network structure (involving users and follow links) before sampling. The evaluation metrics will be introduced in the following subsection.

5) Community Detection Evaluation Metrics: The *community detection* evaluation metrics used in the experiments include, Density, Separability, Coverage, and Expansion, which have been frequently used as the metrics for topological clustering problems. An introduction to these metrics is available in [1] and [20].

C. Link Prediction Experimental Results

In this link prediction task, we compare the performance of five different embedding methods under different *sampling ratio* $\lambda \in \{10\%, 20\%, \cdots, 100\%\}$. We try to predict the follow link relationship in the testing set with the sampled training data. The negative positive rate θ is set with value 1 here (i.e., negative and positive sets are of the same size).

The method we proposed in this paper, DIME, performs much better than the other methods in the link prediction task, since the heterogeneous information from both the emerging and other aligned mature networks used in DIME can provide extra information to help the model learn the embedding feature vectors of the users. Table IV shows the performance of DIME, DIME-SH, and other three baseline methods, Auto-eocoder, LINE and DeepWalk, evaluated by AUC, Accuracy, Recall and F1 with different sampling ratio λs.

When the *sampling ratio* λ is low, like 10%, the baseline models will suffer from the information sparsity a lot, but by transferring information other aligned source networks DIME can still obtain very good performance. As the *sampling ratio* λ increases, the performance of all these methods improves steadily, and DIME can outperform the other methods with great advantages consistently.

Among all the baseline methods, DIME can achieve the best performance in most of the cases (except the Recall measure with $\lambda \in \{10\%, 20\%, 30\%, 40\%\}$). For instance, when $\lambda = 30\%$, the AUC achieved by DIME is 0.838, which is 4.5% higher than the AUC obtained by DIME-SH. It demonstrates our assumption that "information from other aligned networks can help improve the performance greatly". The advantages of DIME will be more significant
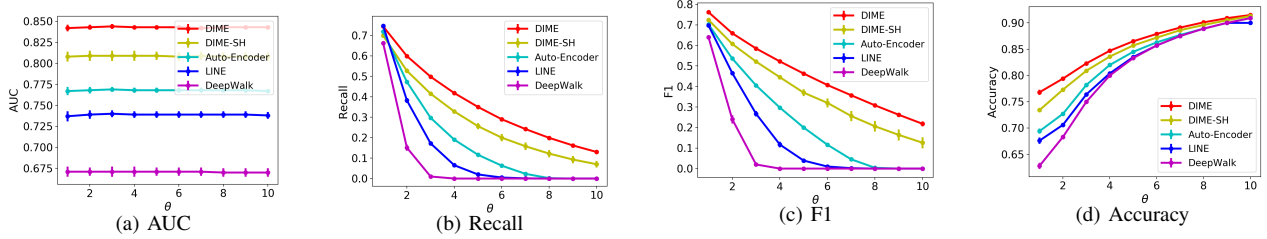
(a) AUC   (b) Recall   (c) F1   (d) Accuracy

Fig. 2. Parameter Analysis of negative positive rate θ in link prediction.

compared with the remaining baseline methods. Meanwhile, with heterogeneous information in the emerging network, DIME-SH can also outperform the other baseline models built with homogeneous information only. For instance, the AUC, Accuracy, and F1 obtained by DIME-SH are all over 8% greater than the measures obtained by Auto-encoder, LINE and DeepWalk. It shows the meta proximity proposed in this paper can effectively capture the network structure information for the users.

D. Link Prediction Parameter Sensitivity Analysis

In the link prediction task, we set the negative positive rate θ equals 1. In this part, we will provide the sensitivity analysis of parameter θ. Figure 2 shows the AUC, Recall, F1 and Accuracy of the comparison methods with negative positive sample rate $\theta \in \{1, 2, \cdots, 10\}$.

As the negative positive rate θ increases, more negative links will be added to the training and testing set, which renders the link prediction task more challenging. According to Figure 2, we observe that, the metrics like Recall and F1 are all decreasing as θ gets larger. The AUC curve is relative stable, as it is not very sensitive to the class imbalance problem. As for Accuracy, when the negative positive rate θ increases, the data gets more imbalanced. In such a class imbalance circumstance, the high Accuracy scores will not be that meaningful.

Through Figure 2, we can observe that even the metrics like Recall and F1 will all degrade as the the negative positive rate θ increase, the decreasing speed of different methods is different. DIME decreases slower than DIME-SH, while the decreasing speed of DIME-SH is much slower than the remaining baseline methods. This means that although the performance of all the methods are influenced by the increasing parameter θ, DIME and DIME-SH are more stable than the other baseline models.

E. Community Detection Experimental Results

In Table V, we show the community detection results obtained by the comparison methods evaluated by Density, Separability, Coverage and Expansion with different sampling ratios $\lambda \in \{10\%, 20\%, \cdots, 100\%\}$. Here the number of cluster, i.e., parameter k, is assigned with value 10, whose sensitivity analysis will be provided in the following subsection.

According to the results shown in the table, we observe that DIME performs the best out of all the methods. As we can see, when the sample ratio increases, the emerging network will have more information, and the community detection results obtained by all the comparison methods will increase steadily.

Under the same sample ratio, the comparison methods sorted in the descending order according to their performance are as follows: DIME, DIME-SH, Auto-Encoder, LINE and DeepWalk. By comparing DIME with the other baseline methods, DIME can outperform them with great advantages. For instance, when sample rate λ equals 0.5, the Separability of DIME is 0.440, which is 41.5% larger than that obtained by DIME-SH. Based on the meta proximity and heterogeneous information in the emerging network, DIME-SH can perform much better than the other homogeneous network based embedding methods. For instance, the Separability achieved by DIME-SH is 20% larger than that of Auto-encoder, and over 70%

greater than LINE and DeepWalk. The results are also very similar for other evaluation metrics.

F. Community Detection Parameter Sensitivity Analysis

In the community detection task, we set the community number k with 10. In this part, we try to analyze how will the performance be influenced while the number of communities k differs. Figure 3 shows the change of Density, Separability, Coverage, Expansion obtained by the comparison methods while k increases from 10 to 100.

Generally, when the community number k increases from 10 to 20, the performance of all the methods degrades a little bit, and when k increases from 20 to 30, the performance increases again, which will keep dropping steadily as k further increases. In the community detection task, we do not know how many communities in there. So we need to try different ks to get best performance. In our case, $k = 10$ achieves best performance among the values in $\{10, 20, \cdots, 100\}$.

If we sort the comparison methods according to their performance in the decreasing order, the sorted list will be DIME, DIME-SH, Auto-encoder, LINE and DeepWalk. The heterogeneous information across the emerging and aligned source networks used in DIME help the clustering model to group similar people together.

V. RELATED WORK

Network embedding has become a very hot research problem in recent years, which can project a graph-structured data to the feature vector representations automatically. In the graphs, the relation can be treated as a translation of the entities, and many translation based embedding models have been proposed. Model TransE [3] is the initial translation based embedding work, which projects the entity and relation into a common feature space. TransH [19] improves TransE by considering the link cardinality constraint in the embedding process, and can achieve comparable time complexity. In the real-world multi-relational networks, the entities can have multiple aspects, and the different relations can express different aspects of the entity. Model TransR [11] proposes to build the entity and relation embeddings in separate entity and relation spaces instead.

In recent years, many recent network embedding works based on random walk model and deep learning models have proposed, like Deepwalk [14], LINE [18], node2vec [6], HNE [4]. Perozzi et al. extends the word2vec [12] to the network scenario and introduce the Deepwalk algorithm [14], which uses local information obtained from truncated random walks to learn latent representations by treating walks as the equivalent of sentences. Tang et al. [18] propose to embed the networks with LINE algorithm, which can preserve both the local and global network structures. An edge-sampling algorithm is applied in LINE that addresses the limitation of the classical stochastic gradient descent and improves both the effectiveness and the efficiency of the inference. Grover et al. [6] introduces a flexible notion of a node's network neighborhood and design a biased random walk procedure to sample the neighbors in the training process, which efficiently explores diverse neighborhoods. Chang et al. [4] learns the embedding of heterogeneous networks involving both text and image information. Chen et al. [5] introduce a task guided embedding model to learning the representations for the author identification problem.
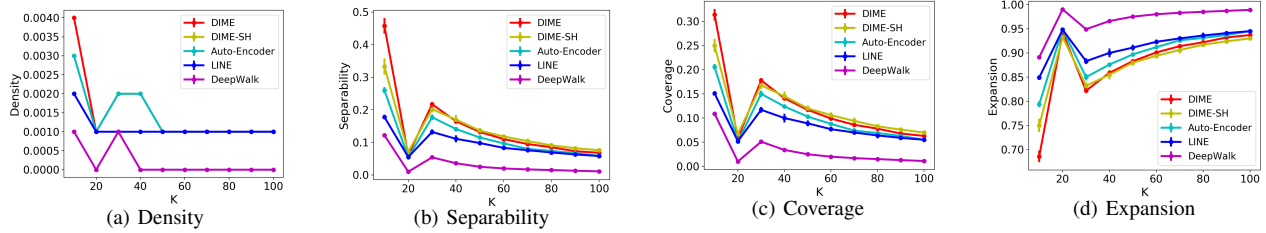
(a) Density   (b) Separability   (c) Coverage   (d) Expansion

Fig. 3. Parameter Analysis of community number k.

Link prediction and recommendation first proposed in [9] has become a very important problem in online social networks, which provides social network researchers with the opportunity to study both the network properties from the individuals social connection perspective. Traditional unsupervised link predictor proposed in [9] mainly calculate the closeness scores among users, and assume that close users tend to be friends in the network. Hasan et al. [7] is the first to study the link prediction problem as a supervised learning problem, where the existing and non-existing social links are treated as the positive and negative instances respectively. Today, many social networks are heterogeneous and to conduct the link prediction in these networks, Sun et al. [17] propose a meta path-based prediction model to predict co-author relationship in the heterogeneous bibliographic network.

Clustering method has also been widely used to detect communities in networks. Newman et al. introduce a modularity function measuring the quality of a division of networks [13]. Shi et al. introduce the concept of normalized cut and discover that the eigenvectors of the Laplace matrix provide a solution to the normalized cut objective function [15]. In addition, many community detection works have been done on heterogeneous online social networks. Sun et al. [16] propose to study the clustering problem with complete link information but incomplete attribute information. Lin et al. [10] try to detect the communities in networks with incomplete relational information but complete attribute information.

VI. CONCLUSION

In this paper, we propose to study the embedding problem for emerging online social networks with broad learning, namely the BL-MNE problem. Emerging networks denote the social networks that newly created containing very little social information. Traditional embedding models will suffer from the information sparsity problem a lot in handling such emerging networks. To solve problem, we introduce a novel embedding framework DIME. Based on a set of meta proximity, DIME can make full use of the heterogeneous information inside the network. Via the cross-network information transfer, DIME refines the embedding results with information from other external aligned mature networks. To demonstrate the effectiveness of DIME, extensive experiments have been done on real-world social networks, which include two main tasks: link prediction and community detection. The experimental results show that DIME can perform very well in learning the embedding vectors for nodes in the emerging networks.

VII. ACKNOWLEDGEMENT

REFERENCES

[1] H. Almeida, D. Guedes, W. Meira, and M. Zaki. Is there a best quality metric for graph clusters? In *ECML PKDD*, 2011.
[2] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *NIPS*, 2006.
[3] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*. 2013.
[4] S. Chang, W. Han, J. Tang, G. Qi, C. Aggarwal, and T. Huang. Heterogeneous network embedding via deep architectures. In *KDD*, 2015.
[5] T. Chen and Y. Sun. Task-guided and path-augmented heterogeneous network embedding for author identification. *CoRR*, abs/1612.02814, 2016.
[6] A. Grover and J. Leskovec. Node2vec: Scalable feature learning for networks. In *KDD*, 2016.
[7] M. Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In *SDM*, 2006.
[8] X. Kong, J. Zhang, and P. Yu. Inferring anchor links across multiple heterogeneous social networks. In *CIKM*, 2013.
[9] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *CIKM*, 2003.
[10] W. Lin, X. Kong, P. Yu, Q. Wu, Y. Jia, and C. Li. Community detection in incomplete information networks. In *WWW*, 2012.
[11] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, 2015.
[12] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.
[13] M. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review*, 2004.
[14] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *KDD*, 2014.
[15] J. Shi and J. Malik. Normalized cuts and image segmentation. *TPAMI*, 2000.
[16] Y. Sun, C. Aggarwal, and J. Han. Relation strength-aware clustering of heterogeneous information networks with incomplete attributes. *VLDB*, 2012.
[17] Y. Sun, R. Barber, M. Gupta, C. Aggarwal, and J. Han. Co-author relationship prediction in heterogeneous bibliographic networks. In *ASONAM*, pages 121–128, 2011.
[18] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *WWW*, 2015.
[19] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, 2014.
[20] Jaewon Yang and J Leskovec. Defining and evaluating network communities based on ground-truth. In *ICDM*, 2012.
[21] 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.
[22] J. Zhang, L. Cui, P. Yu, Y. Lv, and Y. Fu. Bl-ecd: Broad learning based enterprise community detection via hierarchical structure fusion. In *CIKM*, 2017.
[23] J. Zhang, Y. Lv, and P. Yu. Enterprise social link prediction. In *CIKM*, 2015.
[24] J. Zhang and P. Yu. Community detection for emerging networks. In *SDM*, 2015.
[25] J. Zhang and P. Yu. Multiple anonymized social networks alignment. In *ICDM*, 2015.
[26] J. Zhang, P. Yu, and Y. Lv. Organizational chart inference. In *KDD*, 2015.
[27] J Zhang, P. Yu, and Y. Lv. Enterprise community detection. In *ICDE*, 2017.
[28] J. Zhang, P. Yu, and Y. Lv. Enterprise employee training via project team formation. In *WSDM*, 2017.
[29] J. Zhang, P. Yu, Y. Lv, and Q. Zhan. Information diffusion at workplace. In *CIKM*, 2016.
[30] J. Zhang, P. Yu, and Z. Zhou. Meta-path based multi-network collective link prediction. In *KDD*, 2014.
[31] 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.