Label Informed Attributed Network Embedding

Xiao Huang,[†] Jundong Li,[‡] Xia Hu[†,§]
[†]Department of Computer Science and Engineering, Texas A&M University, College Station, TX, USA
[‡]Computer Science and Engineering, Arizona State University, Tempe, AZ, USA
[§]Center for Remote Health Technologies and Systems, Texas A&M Engineering Experiment Station, College Station, TX, USA
{xhuang,xiahu}@tamu.edu, jundongl@asu.edu

ABSTRACT

Attributed network embedding aims to seek low-dimensional vector representations for nodes in a network, such that original network topological structure and node attribute proximity can be preserved in the vectors. These learned representations have been demonstrated to be helpful in many learning tasks such as network clustering and link prediction. While existing algorithms follow an unsupervised manner, nodes in many real-world attributed networks are often associated with abundant label information, which is potentially valuable in seeking more effective joint vector representations. In this paper, we investigate how labels can be modeled and incorporated to improve attributed network embedding. This is a challenging task since label information could be noisy and incomplete. In addition, labels are completely distinct with the geometrical structure and node attributes. The bewildering combination of heterogeneous information makes the joint vector representation learning more difficult. To address these issues, we propose a novel Label informed Attributed Network Embedding (LANE) framework. It can smoothly incorporate label information into the attributed network embedding while preserving their correlations. Experiments on real-world datasets demonstrate that the proposed framework achieves significantly better performance compared with the state-of-the-art embedding algorithms.

1. INTRODUCTION

Attributed networks [18, 28] are ubiquitous in a variety of real-world information systems, such as academic networks and health care systems. Different from plain networks in which only node-to-node interactions and dependencies are observed, each node in an attributed network is often associated with a rich set of features. For instance, with the popularity of social networking services, people not only make friends with each other to form online communities but also actively share opinions and post comments. In social science, social influence theories [22, 23] have been studied that attributes of individuals can both reflect and affect their community structures [46]. In addition, a number of data mining applications, such as sentiment analysis [12] and trust prediction [34], have been benefited by exploiting the correlations between geometrical structure and node attributes. Network embedding [36, 44], as an efficient computational tool for graph mining, aims at mapping the topological proximities of all nodes in a network into a continuous low-dimensional matrix representation. The learned embedding representation paves the way for numerous applications such as node classification [33, 47], link prediction [10, 31], and network visualization [35]. While this has been extensively studied, research on Attributed Network Embedding (ANE) [7] is still in its early stage. In contrast to network embedding that learns from pure networks, ANE targets at leveraging both network proximity and node attribute affinity. Due to heterogeneity of the two information sources, it is challenging for existing network embedding algorithms to be directly applied on ANE.

Abundant labels such as group or community categories have been collected in various real-world networks. For instance, in many social networks such as Facebook and Flickr, users are allowed to join some predefined groups. Users in the same group tend to share posts or photos of similar themes, and they also frequently interact with each other. Citation network is another example. Papers published in the same research community usually share common topics. They also heavily cite others from the same community. These facts can be explained by the homophily hypothesis [16, 21], i.e., individuals with the same label usually have similar social relations and similar node attributes. Labels are strongly influenced by and inherently correlated to both of the network structure and attribute information. Motivated by the fact that labels are potentially helpful in learning a better joint embedding representation, while existing methods focus on the problem in an unsupervised manner, we propose to study how label information can be leveraged and incorporated into ANE.

However, it is a nontrivial task to model and take advantage of labels in attributed networks. There are two main challenges. First, the attributed network and label information could be sparse, incomplete and noisy. For instance, in social networks, the number of single user's friends is always exceedingly limited compared with the total number of users [1]. The proportion of active users who have specified their labels might also be quite small. Second, it is challenging to learn a unified representation given the heterogeneity of an attributed network and its labels. Different from attributes such as comments and posts, labels separate the

WSDM 2017, February 06-10, 2017, Cambridge, United Kingdom

DOI: http://dx.doi.org/10.1145/3018661.3018667

instances into different groups or communities. It is difficult to explicitly model the correlations among all these multimodal information sources, and jointly embed them into an informative embedding representation. Therefore, it is a challenging task to leverage the heterogeneous and noisy information to compensate each other towards an effective and robust embedding representation.

In this paper, we investigate the problem of label informed attributed network embedding and propose an effective framework LANE. Specifically, we aim at answering the following questions: (1) How to model and incorporate labels into an ANE framework? (2) What are the potential impacts of labels on the embedding representation learning? (3) How much can LANE contribute to other learning tasks such as node classification by leveraging labels? The main contributions of this paper are summarized as follows:

- Formally define the problem of label informed attributed network embedding;
- Propose a novel framework LANE, which can affiliate labels with the attributed network and smoothly embed them into a low-dimensional representation by modeling their structural proximities and correlations;
- Present an effective alternating algorithm to solve the optimization problem of LANE; and
- Empirically evaluate and validate the effectiveness of LANE on real-world attributed networks.

2. PROBLEM STATEMENT

Notations: We use lowercase alphabets (e.g., a) to denote scalars, boldface lowercase alphabets (e.g., $\mathbf{a}$) to denote vectors and boldface uppercase alphabets (e.g., $\mathbf{A}$) to denote matrices. The i^{th} row of a matrix $\mathbf{A}$ is denoted as $\mathbf{a}_i$. The transpose of matrix $\mathbf{A}$ is represented as $\mathbf{A}^T$. The trace of $\mathbf{A}$ is taken as $\text{Tr}(\mathbf{A})$ if $\mathbf{A}$ is square. Operation $\|\cdot\|_2$ denotes the Euclidean norm of a vector. $\mathbf{I}$ denotes the identity matrix. The main symbols are listed in Table 1.

Let $\mathcal{G} = \{\mathbf{G},\mathbf{A}\}$ be an attributed network with n interconnected nodes, where $\mathbf{G} \in \mathbb{R}^{n\times n}$ is the weighted adjacency matrix. The $(i,j)^{\text{th}}$ entry of $\mathbf{G}$ represents the link information from node i to node j, and it is defined as 0 if there is no edge. Matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$ collects the attribute information of all n nodes, and its i^{th} row $\mathbf{a}_i$ denotes the m-dimensional attribute vector of node i. Both $\mathbf{G}$ and $\mathbf{A}$ can be either binary or take any real value. In addition to the geometrical structure and node attributes, each node is associated with label information indicating its affiliation group or groups. Let $\mathbf{Y} \in \mathbb{R}^{n\times k}$ be a binary matrix that collects the labels of all n nodes, where k is the number of label categories. $\mathbf{Y}_{ij} = 1$ indicates node i belongs to category j. Each node can belong to one or several categories.

Based on the terminologies described above, we formally define the problem of label informed attributed network embedding as follows: *Given an attributed network $\mathcal{G}$ associated with label information $\boldsymbol{Y}$, we aim at representing each node i as a continuous low-dimensional vector representation $\mathbf{h}_i \in \mathbb{R}^d$, i.e., learning a mapping $f : \{\mathcal{G},\mathbf{Y}\} \to \mathbf{H}$ in a way that agrees with both attributed network and labels as much as possible, such that better performance can be achieved by $\mathbf{H}$ in terms of advancing other learning tasks.*

Notations	Definitions
$\mathbf{G}$	weighted adjacency matrix
$\mathbf{A}$	attribute information matrix
$\mathbf{Y}$	label information matrix
$\mathbf{H}$	final embedding representation
$\mathbf{S}^{(G)}$	network affinity matrix
$\mathbf{S}^{(A)}$	node attribute affinity matrix
n	number of nodes in the network
d	dimension of the embedding representation

Table 1: Main symbols and definitions in the paper.

3. LABEL INFORMED EMBEDDING - LANE

We propose a novel label informed attributed network embedding method - LANE, which can model the node proximities in attributed network space and label information space, as well as jointly embed them into a unified low-dimensional representation. Figure 1 illustrates the main idea of LANE. In the figure, there is an attributed network with six nodes. Each node is associated with specific labels. LANE jointly embeds the attributed network and labels via two modules: attributed network embedding and label informed embedding. First, we map the node proximities in network structure and attribute information into two latent representations $\mathbf{U}^{(G)}$ and $\mathbf{U}^{(A)}$, and then incorporate $\mathbf{U}^{(A)}$ into $\mathbf{U}^{(G)}$ by extracting their correlations. Second, we employ the learned joint proximity to smooth the label information and uniformly embed them into another latent representation $\mathbf{U}^{(Y)}$, and then project all of the learned latent representations into a unified embedding representation $\mathbf{H}$. In this space, nodes 1 and 3 are represented as similar vectors $\begin{bmatrix}0.54 & 0.27\end{bmatrix}$ and $\begin{bmatrix}0.55 & 0.28\end{bmatrix}$, which means that they have similar properties in the original space. To approach the optimal representation efficiently, we also design an effective alternating optimization algorithm. Next, we introduce LANE in detail.

3.1 Attributed Network Embedding Module

In this module, our goal is to seek two $n \times d$ matrices to represent nodes in $\mathcal{G}$ such that structural information of network and attributes is well preserved. Specifically, we aim at allocating similar vector representations for nodes with similar geometrical or attribute proximities respectively.

We first introduce how to model the network structure. The key idea is to focus on each pair of nodes i and j. If they have similar locality properties, then their vector representations $\mathbf{u}_i$ and $\mathbf{u}_j$ should also be similar in the learned space. We use distance $\|\mathbf{u}_i - \mathbf{u}_j\|_2^2$ to measure this. For example, in Figure 1, nodes 1 and 3 share similar geometrical structures since both are connected with nodes 2 and 4, so $\mathbf{u}_2$ and $\mathbf{u}_4$ tend to be similar as well. In this paper, we use cosine measure s_{ij} to calculate the similarity of two nodes, and it is straightforward to extend to other measures. Since s_{ij} would be large if nodes i and j have similar network structures, and approach small value otherwise, we can use the following product to measure the *degree of disagreement* between s_{ij} and $\{\mathbf{u}_i,\mathbf{u}_j\}$:

$$s_{ij}\|\mathbf{u}_i - \mathbf{u}_j\|_2^2. \tag{1}$$

This loss function should be small if we want the degree of disagreement to be small. This can be illustrated by the observations that when nodes i and j are similar, $\mathbf{u}_i$ and $\mathbf{u}_j$ should be close to each other. When they are dissimilar,

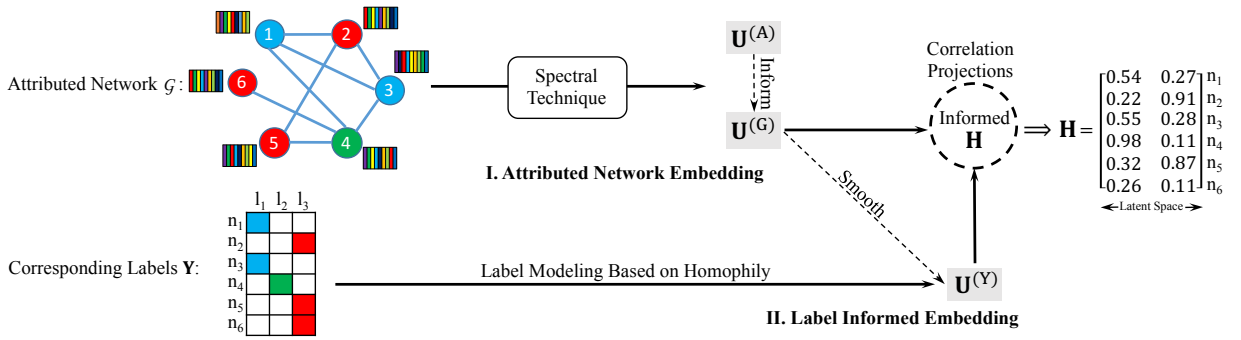

Figure 1: LANE maps the node proximities in attributed network into $\mathbf{U}^{(G)}$ and label proximity into $\mathbf{U}^{(Y)}$, as well as incorporates them into a joint embedding representation H via correlation projections.

another term s_{ij} would be small. Therefore, mathematically we can measure the total degree of disagreement by summing up the product of pairwise similarity and corresponding vector representations' distance, which forms as

$$\underset{\mathbf{U}^{(G)}}{\text{minimize}} \quad \frac{1}{2}\sum_{i,j=1}^{n} s_{ij} \| \frac{\mathbf{u}_i}{\sqrt{d_i}} - \frac{\mathbf{u}_j}{\sqrt{d_j}} \|_2^2. \tag{2}$$

$\mathbf{u}_i$ and $\mathbf{u}_j$ are the i^{th} and j^{th} rows of network latent representation $\mathbf{U}^{(G)}$. We represent the pairwise similarities as a graph affinity matrix $\mathbf{S}^{(G)}$, where s_{ij} is its $(i,j)^{\text{th}}$ element. d_i and d_j are the sum of i^{th} and j^{th} rows of $\mathbf{S}^{(G)}$. We utilize them for normalization purpose. Based on the definition of normalized graph Laplacian [8], we can reformulate Eq. (2) into a maximization problem, and model the geometrical proximity via the objective function as follows,

$$\begin{aligned} \underset{\mathbf{U}^{(G)}}{\text{maximize}} \quad & \mathcal{J}_G = \text{Tr}(\mathbf{U}^{(G)^T} \mathcal{L}^{(G)} \mathbf{U}^{(G)}) \\ \text{subject to} \quad & \mathbf{U}^{(G)^T} \mathbf{U}^{(G)} = \mathbf{I}. \end{aligned} \tag{3}$$

Laplacian $\mathcal{L}^{(G)} = \mathbf{D}^{(G)^{-\frac{1}{2}}} \mathbf{S}^{(G)} \mathbf{D}^{(G)^{-\frac{1}{2}}}$, and degree matrix $\mathbf{D}^{(G)}$ is a diagonal matrix with sum of each row of $\mathbf{S}^{(G)}$ on the diagonal. A constraint is added to avoid being arbitrary.

Similar to the modeling of network structure, we perform the same procedure on node attribute proximity to obtain the attribute latent representation $\mathbf{U}^{(A)}$. We also use cosine similarity to construct the attribute affinity matrix $\mathbf{S}^{(A)}$, and aim at minimizing the degree of disagreement between $\mathbf{U}^{(A)}$ and $\mathbf{S}^{(A)}$. We denote the corresponding Laplacian as $\mathcal{L}^{(A)} = \mathbf{D}^{(A)^{-\frac{1}{2}}} \mathbf{S}^{(A)} \mathbf{D}^{(A)^{-\frac{1}{2}}}$, where $\mathbf{D}^{(A)}$ is the degree matrix of $\mathbf{S}^{(A)}$. Then the objective function of node attributes embedding is defined as

$$\begin{aligned} \underset{\mathbf{U}^{(A)}}{\text{maximize}} \quad & \mathcal{J}_A = \text{Tr}(\mathbf{U}^{(A)^T} \mathcal{L}^{(A)} \mathbf{U}^{(A)}) \\ \text{subject to} \quad & \mathbf{U}^{(A)^T} \mathbf{U}^{(A)} = \mathbf{I}. \end{aligned} \tag{4}$$

To incorporate $\mathbf{U}^{(A)}$ into $\mathbf{U}^{(G)}$, we project $\mathbf{U}^{(A)}$ into the space of $\mathbf{U}^{(G)}$, and employ variance of the projected matrix as a measurement of the correlations [15], i.e.,

$$\rho_1 = \text{Tr}(\mathbf{U}^{(A)^T} \mathbf{U}^{(G)} \mathbf{U}^{(G)^T} \mathbf{U}^{(A)}). \tag{5}$$

By collectively maximizing $\mathcal{J}_G$, $\mathcal{J}_A$, and their correlations, we are able to make $\mathbf{U}^{(A)}$ and $\mathbf{U}^{(G)}$ compensate each other.

In this module, we perform the ANE based on pairwise similarities. The proposed solution is in line with spectral clustering [4, 26], which enjoys several nice properties as follows. First, the spectral technique does not have strong assumptions on the inputs, which means that it can be generalized to many real-world problems. Second, the objective function can be nicely interpretable with many graph theories, such as ratio-cut partitioning [3] and random walks [41]. Third, the corresponding optimization can be readily implemented via eigen-decomposition [8].

3.2 Label Informed Embedding Module

Label information plays an essential role in determining the inscape of each node with strong intrinsic correlations to network structure and node attributes. Along with these strong correlations, labels can potentially be incorporated into the proposed ANE module in Section 3.1. However, labels are usually noisy and incomplete. Direct exploration of labels may negatively affect final embedding results. We propose a principled way to model labels and reinforce the embedding representation learning in two steps: label information modeling and correlation projections.

3.2.1 *Label Information Modeling*

In this step, we map the node proximities in labels into a latent representation $\mathbf{U}^{(Y)}$. The basic idea is to employ the learned attribute network proximity to smooth label information modeling. When nodes have the same label, their geometrical structures, attribute affinity, and final vector representations tend to be similar [21, 23].

Specifically, we include nodes with the same label into the same clique, and the corresponding form of presentation is formulated as $\mathbf{Y}\mathbf{Y}^T$. Based on this, we perform the label proximity modeling. Let $\mathbf{S}^{(YY)}$ be the cosine similarity of $\mathbf{Y}\mathbf{Y}^T$. Similar to $\mathbf{S}^{(G)}$ and $\mathbf{S}^{(A)}$, matrix $\mathbf{S}^{(YY)}$ could be considered as an affinity matrix of label information. We calculate the Laplacian via $\mathcal{L}^{(YY)} = \mathbf{D}^{(Y)^{-\frac{1}{2}}} \mathbf{S}^{(YY)} \mathbf{D}^{(Y)^{-\frac{1}{2}}}$, where $\mathbf{D}^{(Y)}$ is the degree matrix of $\mathbf{S}^{(YY)}$.

However, due to the special structure, the rank of matrix $\mathbf{S}^{(YY)}$ is limited by the number of label categories k, which might be smaller than the embedding dimension d. It leads to unsatisfactory performance of the eigen-decomposition of $\mathcal{L}^{(YY)}$. To address this issue, we utilize the learned proximity $\mathbf{U}^{(G)}\mathbf{U}^{(G)^T}$ to smooth the modeling, and leverage the following objective function to drive nodes with same labels to have similar vector representations,

$$\begin{aligned} \underset{\mathbf{U}^{(Y)}}{\text{maximize}} \quad & \mathcal{J}_Y = \text{Tr}\left(\mathbf{U}^{(Y)^T}(\mathcal{L}^{(YY)} + \mathbf{U}^{(G)}\mathbf{U}^{(G)^T})\mathbf{U}^{(Y)}\right) \\ \text{subject to} \quad & \mathbf{U}^{(Y)^T}\mathbf{U}^{(Y)} = \mathbf{I}. \end{aligned} \tag{6}$$

There are several advantages of doing this. First, Hermitian matrix $\mathbf{U}^{(G)}\mathbf{U}^{(G)^T}$ is in a low-rank space where noise is significantly reduced [41]. Second, the joint proximity is considered to be informative and in line with node proximities in labels. The second term $\mathrm{Tr}(\mathbf{U}^{(Y)^T}\mathbf{U}^{(G)}\mathbf{U}^{(G)^T}\mathbf{U}^{(Y)})$ also measures the correlation between $\mathbf{U}^{(G)}$ and $\mathbf{U}^{(Y)}$. It is beneficial to the label proximity learning since they are considered to be highly correlated [16]. Third, the noise in the learned latent representation $\mathbf{U}^{(Y)}$ could also be greatly reduced [24] and most information in the original label space is recoverable [41]. Therefore, although label information might be incomplete and noisy, we are still able to fully capture the proximities of nodes in the label space.

3.2.2 Correlation Projections

We have embedded attributed network and labels into latent representations $\mathbf{U}^{(A)}$, $\mathbf{U}^{(G)}$ and $\mathbf{U}^{(Y)}$. Next, we design a correlation projection schema to jointly project them into a unified embedding representation $\mathbf{H}$.

Specifically, since all of the latent representations are constrained by corresponding Laplacian, we project all of them into a new space $\mathbf{H}$ to get more degree of freedom and flexibility. To preserve the information in $\mathbf{U}^{(G)}$, we leverage the variance of the projected matrix as a metric of their correlations, which is defined as

$$\rho_2 = \mathrm{Tr}(\mathbf{U}^{(G)^T}\mathbf{H}\mathbf{H}^T\mathbf{U}^{(G)}). \tag{7}$$

Similarly, we project $\mathbf{U}^{(A)}$ and $\mathbf{U}^{(Y)}$ into the space of $\mathbf{H}$ and measure their correlation as

$$\rho_3 = \mathrm{Tr}(\mathbf{U}^{(A)^T}\mathbf{H}\mathbf{H}^T\mathbf{U}^{(A)}), \text{ and} \tag{8}$$

$$\rho_4 = \mathrm{Tr}(\mathbf{U}^{(Y)^T}\mathbf{H}\mathbf{H}^T\mathbf{U}^{(Y)}). \tag{9}$$

The loss function for entire three projections is defined as

$$\underset{\mathbf{U}^{(\cdot)},\mathbf{H}}{\text{maximize}} \quad \mathcal{J}_{corr} = \rho_2 + \rho_3 + \rho_4, \tag{10}$$

where $\mathbf{U}^{(\cdot)}$ denotes all three latent representations. By maximizing ρ_2, ρ_3 and ρ_4 simultaneously, we are able to learn the correlations among $\mathbf{U}^{(G)}$, $\mathbf{U}^{(A)}$, $\mathbf{U}^{(Y)}$ and $\mathbf{H}$, and drive $\mathbf{H}$ to be jointly advanced by the node proximities in attributed network and label information.

3.3 Joint Representation Learning via LANE

We have separately implemented and formulated attributed network embedding and label informed embedding modules. We define two parameters to weight the importance of different measurements and combine them as

$$\begin{aligned} \underset{\mathbf{U}^{(\cdot)},\mathbf{H}}{\text{maximize}} \quad & \mathcal{J} = (\mathcal{J}_G + \alpha_1\mathcal{J}_A + \alpha_1\rho_1) + \alpha_2\mathcal{J}_Y + \mathcal{J}_{corr} \\ \text{subject to} \quad & \mathbf{U}^{(G)^T}\mathbf{U}^{(G)} = \mathbf{I}, \quad \mathbf{U}^{(A)^T}\mathbf{U}^{(A)} = \mathbf{I}, \\ & \mathbf{U}^{(Y)^T}\mathbf{U}^{(Y)} = \mathbf{I}, \quad \mathbf{H}^T\mathbf{H} = \mathbf{I}, \end{aligned} \tag{11}$$

where α_1 is a positive parameter that balances the contribution of attributes in the ANE module. α_2 is a positive parameter that makes a trade-off between the ANE and label informed embedding. By seeking an optimal value of $\mathcal{J}$, we are able to make the embedding representation learning and correlation projections highly relevant and mutually interrelated. In this way, $\mathbf{H}$ is able to capture all of the structural proximities and their correlations in the label informed attributed network.

Algorithm 1: Label informed Attributed Network Embedding

Input: d, ϵ, $\mathcal{G}$, $\mathbf{Y}$.
Output: Embedding representation $\mathbf{H}$.

```
Construct the affinity matrices S^(G) and S^(A);
Compute Laplacian matrices L^(G), L^(A) and L^(Y);
Initialize t = 1, U^(A) = 0, U^(Y) = 0 and H = 0;
repeat
    Update U^(G) by solving Eq. (13);
    Update U^(A) by solving Eq. (14);
    Update U^(Y) by solving Eq. (15);
    Update H by solving Eq. (16);
    t = t + 1;
until J_t − J_{t−1} ≤ ε;
return H.
```

3.4 Optimization Algorithm for LANE

We propose an effective algorithm to solve the problem in Eq. (11). There are four variable matrices, and it is infeasible to give closed-form solutions. Motivated by previous studies [6, 17], we employ an alternating algorithm to approach the optimal status. The key idea is to update a local maximum solution for one of the four variable matrices while fixing others. Eq. (11) would be converted to a convex problem w.r.t. one variable matrix when other three are constant. We now introduce the updating steps in detail.

The second order derivative of $\mathcal{J}$ w.r.t. $\mathbf{U}^{(G)}$ is formed as

$$\nabla^2_{\mathbf{U}^{(G)}}\mathcal{J} = \mathcal{L}^{(G)} + \alpha_1\mathbf{U}^{(A)^T}\mathbf{U}^{(A)} + \alpha_2\mathbf{U}^{(Y)^T}\mathbf{U}^{(Y)} + \mathbf{H}^T\mathbf{H}, \tag{12}$$

where $\mathcal{L}^{(G)}$ is a symmetrical similarity matrix. Since parameters α_1 and α_2 are positive, as well as $\mathbf{U}^{(Y)^T}\mathbf{U}^{(Y)}$, $\mathbf{U}^{(A)^T}\mathbf{U}^{(A)}$, and $\mathbf{H}^T\mathbf{H}$ are all Hermitian matrices, the second order derivative should always be positive semidefinite. When $\mathbf{U}^{(A)}$, $\mathbf{U}^{(Y)}$ and $\mathbf{H}$ are fixed, Eq. (11) becomes convex w.r.t. $\mathbf{U}^{(G)}$, and we are able to obtain the optimal solution via Lagrange multipliers method. Let $\boldsymbol{\lambda}_i (i = 1, \ldots, 4)$ denote the Lagrange multipliers of four variable matrices respectively. By setting the derivative of Lagrangian $\nabla_{\mathbf{U}^{(G)}}\mathscr{L}$ equal to zero, we have

$$(\mathcal{L}^{(G)} + \alpha_1\mathbf{U}^{(A)}\mathbf{U}^{(A)^T} + \alpha_2\mathbf{U}^{(Y)}\mathbf{U}^{(Y)^T} + \mathbf{H}\mathbf{H}^T)\mathbf{U}^{(G)} = \boldsymbol{\lambda}_1\mathbf{U}^{(G)}, \tag{13}$$

and the solution is corresponding to the top d eigenvectors.

Similarly, it is easy to check that the second order derivatives of $\mathcal{J}$ w.r.t. $\mathbf{U}^{(A)}$, $\mathbf{U}^{(Y)}$ and $\mathbf{H}$ are all guaranteed to be positive semidefinite. Therefore, when only one of the matrices $\mathbf{U}^{(A)}$, $\mathbf{U}^{(Y)}$ and $\mathbf{H}$ is variable, maximizing $\mathcal{J}$ would result in finding the top d eigenvectors of the following problems,

$$(\alpha_1\mathcal{L}^{(A)} + \alpha_1\mathbf{U}^{(G)}\mathbf{U}^{(G)^T} + \mathbf{H}\mathbf{H}^T)\mathbf{U}^{(A)} = \boldsymbol{\lambda}_2\mathbf{U}^{(A)}, \tag{14}$$

$$(\alpha_2\mathcal{L}^{(YY)} + \alpha_2\mathbf{U}^{(G)}\mathbf{U}^{(G)^T} + \mathbf{H}\mathbf{H}^T)\mathbf{U}^{(Y)} = \boldsymbol{\lambda}_3\mathbf{U}^{(Y)}, \tag{15}$$

$$(\mathbf{U}^{(G)}\mathbf{U}^{(G)^T} + \mathbf{U}^{(A)}\mathbf{U}^{(A)^T} + \mathbf{U}^{(Y)}\mathbf{U}^{(Y)^T})\mathbf{H} = \boldsymbol{\lambda}_4\mathbf{H}. \tag{16}$$

In summary, the alternating algorithm for optimizing the function in Eq. (11) is presented in Algorithm 1. Since each updating step is to solve a convex problem, it is guaranteed to converge to a local optimal point [6]. We start the updating from the main information source representation $\mathbf{U}^{(G)}$ in order to have an appropriate initialization. Four

variable matrices are updated according to the local eigen-decomposition equations until the increase of objective function $\mathcal{J}$ falls below the convergence threshold ϵ.

3.5 Complexity Analysis

The proposed framework LANE needs a small number of iterations before it converges. In each iteration, LANE conducts four eigen-decompositions. To calculate the leading d eigenvectors of an $n \times n$ matrix, the time complexity of standard iterative eigensolvers such as Lanczos method [39] is $\mathcal{O}(dn^2)$ in the worst case. Let T_{a} stands for the number of operations required to obtain all of the affinity matrices, then the total time complexity of LANE is $\mathcal{O}(T_{\text{a}} + dn^2)$, which is the same as the complexity of spectral embedding [4]. Since $d \ll n$, the time complexity of LANE should be $\mathcal{O}(n^2 + n\mathcal{T})$, where $\mathcal{T}$ denotes the total number of non-zeros in matrices $\mathbf{G}$ and $\mathbf{A}$. It is also easy to check that the space complexity of LANE is $\mathcal{O}(n^2)$.

3.6 Extensions

Attributes and labels might be unavailable in some networks. For instance, when mobile communication companies want to analyze their customers' network in order to provide better service, they may only be able to collect the contact network and partial label information. Attributes like call contents or personal preferences might not be available. LANE can also handle these cases, where either or both attribute and label information is missing.

We use the case when labels are missing as an illustrated example. In this case, two terms in $\mathcal{J}$ would become unavailable. The first one is "$\mathcal{J}_Y$", which is used to perform label information modeling. The second one is "ρ_4", which is used to model the correlations between label proximities and final embedding representation. Hence, we remove them from $\mathcal{J}$, and rewrite the objective function as

$$\begin{aligned} \underset{\mathbf{U}^{(G)},\mathbf{U}^{(A)},\mathbf{H}}{\text{maximize}} \quad & \mathcal{J}_G + \beta_1\mathcal{J}_A + \beta_2\rho_1 + \rho_2 + \rho_3 \\ \text{subject to} \quad & \mathbf{U}^{(G)^T}\mathbf{U}^{(G)} = \mathbf{I}, \quad \mathbf{U}^{(A)^T}\mathbf{U}^{(A)} = \mathbf{I}, \\ & \mathbf{H}^T\mathbf{H} = \mathbf{I}. \end{aligned} \tag{17}$$

Two positive parameters β_1 and β_2 are introduced to determine the contributions of attributes and correlations. We denote this variation as *LANE_w/o_Label*. The optimal solution can be obtained via a similar alternating algorithm as Algorithm 1, so we omit the details. It is straightforward to extend LANE to the case that attributes are missing.

4. EXPERIMENTS

In this section, we conduct experiments to validate the effectiveness and efficiency of LANE on real-world attributed networks. In particular, we want to answer questions as follow. (1) How effective is the embedding representation learned by LANE compared with representations learned by the state-of-the-art methods in other learning tasks such as node classification? (2) What are the impacts of labels for the ANE? (3) How efficient is the proposed framework compared with other ANE methods?

4.1 Datasets

We first introduce the applied datasets. Two real-world social media datasets BlogCatalog and Flickr are used in our study. The detailed information is shown in Table 2.

	# Nodes	# Edges	# Attributes	# Labels
BlogCatalog	5,196	171,743	8,189	6
Flickr	7,575	239,738	12,047	9

Table 2: Detailed information of the datasets.

Both datasets are publicly available and have been used in previous work [19]. Descriptions are presented as follows.

BlogCatalog is an online community that people can post blogs. Bloggers follow each other and form a network. We use the keywords in bloggers' blog descriptions as the attribute information. The labels are selected from some predefined categories, which indicate bloggers' interests.

Flickr is an image and video hosting website, where users interact with each other via photo sharing. We use the following relationships among users to form a network. Each user can specify a list of tags of interest, which are considered as his/her attribute information. We set the groups that users joined as labels.

4.2 Baselines

LANE is measured against several state-of-the-art embedding algorithms and two variations. They can be separated into four categories. First, to investigate the impact of low-dimensional representation learning, we compare LANE with Original Features. Second, to evaluate the contribution of attributes, three network embedding algorithms are considered, including DeepWalk, LINE, and LANE_on_Net. Third, to understand the impact of label informed embedding, two ANE methods are included, i.e., LCMF and LANE_w/o_Label. Fourth, to analyze the effectiveness of LANE, we compare it with one straightforward method SpecComb, and one multi-view learning method MultiView. The detailed descriptions are listed as follows.

- *Original Features*: It combines original features of network structure and node attributes by concatenating them together. The concatenating original feature space is used for both training and test group.
- *DeepWalk* [27]: It employs truncated random walks on the plain graph and involves language modeling techniques, i.e., word2vec, to analyze the walking tracks.
- *LINE* [36]: It is one of the state-of-the-art embedding algorithms for large-scale networks. It preserves both first and second-order proximities between the nodes.
- *LCMF* [47]: It conducts a joint matrix factorization on the linkage and attribute information, and maps them into a shared subspace. It uses this subspace as the learned representation.
- *SpecComb*: It concatenates the attributed network $\mathcal{G}$ and labels $\mathbf{Y}$ into one matrix, and performs normalized spectral embedding [41] on this combined matrix. The corresponding top d eigenvectors are collected as the embedding representation.
- *MultiView* [17]: It considers the network, attributes, and labels as three views, and applies co-regularized spectral clustering on them collectively.
- *LANE_on_Net* and *LANE_w/o_Label*: They are two variations of LANE, which have been described in Section 3.6. The former one is for a plain network. The latter one only leverages the attributed network, without the help of label informed embedding.

	BlogCatalog					Flickr				
	1/16	1/8	1/4	1/2	1	1/16	1/8	1/4	1/2	1
DeepWalk	0.5488	0.7000	0.7824	0.7937	0.8100	0.3438	0.4597	0.5818	0.6819	0.7382
LINE	0.6663	0.7255	0.7332	0.6959	0.6931	0.3587	0.4920	0.5733	0.6500	0.6413
LANE_on_Net	0.6553	0.6985	0.7590	0.8046	0.8126	0.4298	0.5063	0.5698	0.6300	0.7319
LCMF	0.7119	0.7920	0.8366	0.8646	0.8401	0.3531	0.5065	0.5884	0.7026	0.7381
LANE_w/o_Label	0.7638	0.7977	0.8361	0.8513	0.8685	0.5426	0.6046	0.6578	0.6809	0.8300
SpecComb	0.6138	0.6027	0.6360	0.6965	0.5895	0.4618	0.4943	0.5800	0.7054	0.7816
MultiView	0.6478	0.7046	0.8207	0.8078	0.7903	0.4942	0.4856	0.5843	0.5870	0.8061
LANE	**0.8065**	**0.8523**	**0.8856**	**0.8964**	**0.9008**	**0.6658**	**0.7645**	**0.8267**	**0.8276**	**0.9054**

Table 3: Classification performance (F_1 score) of different methods on different datasets with $d = 100$.

4.3 Experimental Settings

Following a commonly adopted way [27, 36], we validate the effectiveness of different learned representations on node classification task [33, 47]. This task is to predict which category or categories a new node belongs to based on the model learned from training data. We follow the suggestions of original papers to set the parameters of baseline methods. Our proposed method is flexible in choosing the weight parameters α_1 and α_2. We will further discuss the impact of these parameters in Section 4.6.

We employ 5-fold cross-validation in the following experiments. Specifically, we randomly separate all of the n nodes into training group ($\mathbf{G}_{\text{train}}$, $\mathbf{A}_{\text{train}}$, $\mathbf{Y}_{\text{train}}$) and test group ($\mathbf{G}_{\text{test}}$, $\mathbf{A}_{\text{test}}$, $\mathbf{Y}_{\text{test}}$). $\mathbf{G}_{\text{train}}$ is a square matrix since the information of test group is unavailable when training the model. $\mathbf{G}_{\text{test}}$ contains the link information from test group to all other nodes. The goal is to predict the label or labels of each instance k in the test group, while only given its network feature $\mathbf{g}_k$ (i.e., k^{th} row of $\mathbf{G}_{\text{test}}$) and attributes $\mathbf{a}_k$ (i.e., k^{th} row of $\mathbf{A}_{\text{test}}$). To evaluate the performance of a method, we first apply it on training group to obtain the embedding representation $\mathbf{V}_{\text{train}}$. Then we build a Support Vector Machine (SVM) classifier with $\mathbf{V}_{\text{train}}$ and the corresponding labels $\mathbf{Y}_{\text{train}}$. To obtain the embedding representation of test group while eliminating the dependence among test nodes, we first construct two linear mapping functions $\mathbf{B}^{(\mathbf{G})}$ and $\mathbf{B}^{(\mathbf{A})}$, which enable us to map $\mathbf{G}$ and $\mathbf{A}$ into the embedding space $\mathbf{V}$. Mathematically, it forms as

$$\mathbf{G} = \mathbf{V} \cdot \mathbf{B}^{(G)}, \text{ and } \mathbf{A} = \mathbf{V} \cdot \mathbf{B}^{(A)}. \tag{18}$$

We use this linear mapping for the sake of simplicity, but it can be easily extended to other nonlinear mapping functions. Functions $\mathbf{B}^{(\mathbf{G})}$ and $\mathbf{B}^{(\mathbf{A})}$ can be learned based on the training group. Then for the k^{th} node in test group, we can calculate its embedding vector representation $\mathbf{v}_k$ via the following formulation

$$\mathbf{v}_k = \mathbf{g}_k \cdot (\mathbf{B}^{(G)})^{\dagger} + \delta \mathbf{a}_k \cdot (\mathbf{B}^{(A)})^{\dagger}, \tag{19}$$

where δ is a positive weight, and it could be tuned to balance the contributions of $\mathbf{G}$ and $\mathbf{A}$. Function $(\cdot)^{\dagger}$ is the pseudoinverse. At last, we perform the prediction based on the test group embedding representation $\mathbf{V}_{\text{test}}$ and learned SVM classifier.

In the experimental settings, all methods have the access to all three types of information sources, such that we are able to focus on investigating the impact of embedding. We report the node classification performance in terms of F_1 score [27, 36]. The representation dimension d is set to be 100 for all methods. Further experiments have been done to investigate the impact of d in Section 4.6. All experimental results are arithmetic average of 10 trials.

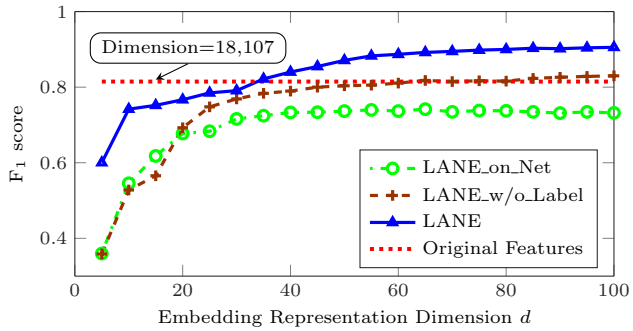

Figure 2: Classification performance of Original Features, LANE and its variations on Flickr dataset.

4.4 Performance Evaluation

To answer the first question proposed in the beginning of Section 4, we evaluate the performance of LANE by comparing it with all baseline methods on the node classification task. Next, we introduce the experimental results in detail.

4.4.1 *Impact of Embedding Representation Learning*

To investigate the effectiveness of low-dimensional embedding representation learning, we compare LANE and its variations with Original Features on both datasets. The dimensions of original feature space on BlogCatalog and Flickr datasets are 12,346 and 18,107 respectively. In the experiments, we vary the embedding representation dimension d from 5 to 100. The classification performance of different methods w.r.t. d on Flickr is presented in Figure 2. We omit the result on BlogCatalog since it is similar.

Experimental results in Figure 2 show that LANE_w/o_Label could achieve similar classification accuracy as Original Features when d is quite small ($d \geq 60$). LANE achieves higher F_1 score than Original Features when $d \geq 35$. This is much lower than the dimension of original feature space 18,107. In addition, LANE_on_Net performs 10.17% worse than Original Features since it only embeds network structure. Therefore, by taking advantage of embedding representation learning, the proposed method LANE achieves better performance than Original Features.

4.4.2 *Effectiveness of LANE*

To study the effectiveness of LANE, we compare its performance with all baseline methods. We fix $d = 100$ and vary the number of instances used for embedding as $\{\frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1\}$ of entire training group. The comparison results are presented in Table 3. From the Table, we observe that LANE always outperforms all baselines among all range of training percentages on both datasets. For instance, on Flickr dataset, by taking advantage of embedding attributes and labels, LANE outperforms LINE at least 41.18%. With the

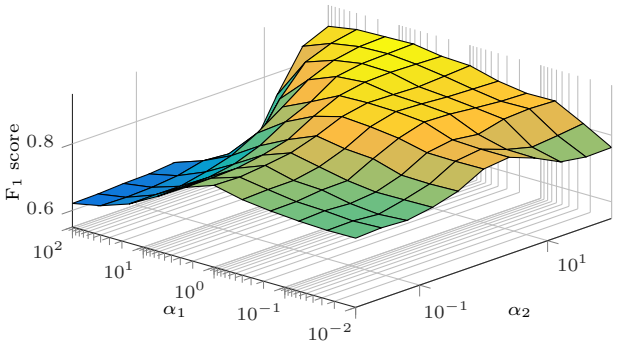

Figure 3: Performance of LANE on Flickr with different parameters α_1 and α_2.

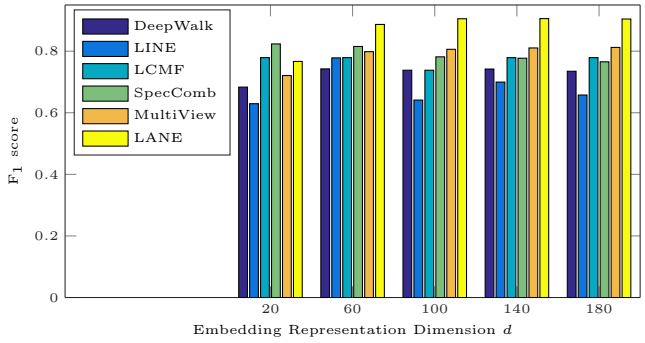

Figure 4: Node classification performance of different methods w.r.t. d on Flickr dataset.

help of label informed embedding, LANE also consistently achieves higher F_1 score than the state-of-the-art ANE method LCMF at least 17.79%, and MultiView at least 12.32%. Meanwhile, when training percentage increases from $\frac{1}{16}$ to 1, the performance of LANE keeps increasing, but growths become smaller. The improvements from the incorporations of ANE and label embedding also become smaller when the training percentage approaching 100%.

We also perform one-tailed t-test between LANE and these baseline methods. Results show that LINAE is significantly better (with a 0.01 significance level). The p-value on BlogCatalog is smaller than 5.27×10^{-16}, and the p-value on Flickr is smaller than 1.61×10^{-14}.

4.5 Impact of Label Informed Embedding

In this subsection, we answer the second question that what is the impact of label information on the embedding representation learning. We analyze it from three aspects.

First, we compare LANE with its own variations. Method LANE_w/o_Label can capture the node proximities in network structure and attributes as well as their correlations. Results in Table 3 show that it obtains better performance than all network embedding methods, including DeepWalk, LINE, and LANE_on_Net. For example, on Flickr, it achieves 12.44% improvement than DeepWalk. By taking advantage of label informed embedding, LANE further outperforms LANE_w/o_Label significantly. As shown in Table 3, LANE advances the performance greatly by 9.08%. This demonstrates the advantage of incorporating labels into the embedding representation learning.

Second, we compare LANE with the ANE method LCMF. In Table 3, we observe that, by affiliating labels with the attributed network, LANE achieves better classification performance than LCMF. For instance, on BlogCatalog, it consistently achieves at least 3.68% improvement than LCMF. This verifies the effectiveness of LANE in incorporating labels into the learned embedding representation.

Third, we compare LANE with SpecComb and MultiView, where SpecComb is a straightforward method. Table 3 shows that SpecComb and MultiView are always inferior to LANE, and sometimes even perform worse than pure network embedding methods such as DeepWalk. The reason is that SpecComb does not explicitly consider the inherent correlations, and concatenation is not an appropriate way of combining heterogeneous information. MultiView considers network structure, node attributes, and labels equally, without capturing their distinct characteristics. Therefore, LANE is an appropriate method for the label incorporation.

All these observations demonstrate the existence of strong correlations between the attributed network and labels. The exploitation of embedding labels indeed could help us obtain a better latent representation, however, an appropriate method is required for the incorporation. LANE successfully achieves this improvement by performing the label informed embedding, and consistently outperforms all baseline methods. We want to emphasize that in the node classification task, all methods have access to the labels of training group and utilize them in different ways. Only LANE, SpecComb, and MultiView can incorporate these labels into embedding representation learning. Therefore, the superiority of LANE is not a result of owning extra information source, but performing label informed embedding.

4.6 Parameter Analysis

In this subsection, we investigate the effects of parameters α_1, α_2 and d. The two weights α_1 and α_2 balances the contributions of attributes and labels respectively. We vary them from 0.01 to 100 simultaneously, and the result on Flickr is presented in Figure 3. A similar result is observed on BlogCatalog, so we omit it. From the figure, we observe that LANE achieves relatively high performance when both attributes and labels have sufficient contributions, i.e., $\alpha_1 > 1$ and $\alpha_2 > 10$. When α_1 is small, the highest performance is achieved when α_2 is in the middle, i.e., $\alpha_2 \approx 1$. A similar observation is made when α_2 is small. When α_1 is fixed to 100, as α_2 increases from 0.01 to 100, there is a significant 42.70% improvement in the performance. When $\alpha_2 = 100$, only 16.07% improvement is observed as α_1 increased. This demonstrates that labels have a larger impact than attributes on LANE. As a conclusion, LANE could achieve relatively high performance by setting reasonable parameters. Significant and positive impacts are observed from α_2, which verify that label informed embedding plays an essential role in the proposed framework.

To study how d affect the node classification performance, we also vary it from 20 to 180. The F_1 score of different methods w.r.t. d is shown in Figures 4. We omit the result on BlogCatalog since it is similar. As we can see, all of the observations mentioned above hold undeviatingly when $d > 20$. For example, LANE consistently achieves higher accuracy than all baseline methods when d varies widely. In addition, we observe that, by increasing d, the classification performance of LANE first increases and then keeps stable. This is consistent on both BlogCatalog and Flickr. It is appealing in real-world applications since practitioners can safely tune these parameters in a wide range.

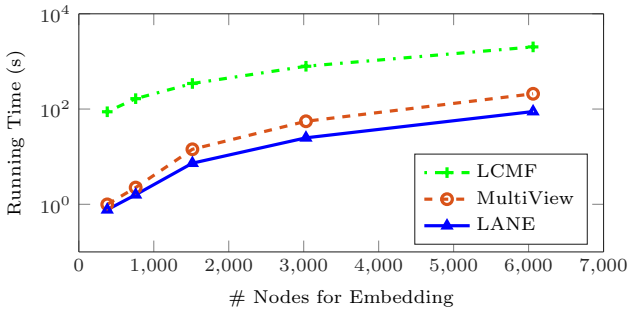

Figure 5: Running time of LCMF, MultiView, and LANE w.r.t. the number of input nodes on Flickr.

4.7 Efficiency Evaluation

To answer the last question asked in the beginning of Section 4, we compare the running time of LANE and two state-of-the-art ANE methods, i.e., LCMF and MultiView. The computation time in logarithmic scale w.r.t. the number of input nodes on Flickr is presented in Figure 5. A similar result is obtained on BlogCatalog. From the results, we can see that LANE takes less running time than all others. This can be explained by the fact that LCMF employs gradient descent in the optimization, which often has slow convergence rate. MultiView has the same time complexity as LANE, but empirical results show that LANE could converge in a few iterations on both datasets. Therefore, Figure 5 demonstrates the efficiency of the proposed method.

5. RELATED WORK

Network embedding enjoys increasing popularity in recent years. Its pioneer work can be traced back to the graph embedding problem, which was introduced by Filotti et al. [9] as a graph genus determining problem in 1979. A family of more general graph embedding approaches [26, 38] were developed around the 2000s. They target at generating low-dimensional manifolds which can model the nonlinear geometry of data, including Isomap [38], Laplacian Eigenmaps [4] and spectral techniques [3, 8]. Up till now, due to the pervasiveness of networked data, a variety of network embedding algorithms [30, 36, 44] have been implemented. Iwata et al. [13] applied probabilistic latent semantic analysis to embed document networks. Tang et al. [37] investigated the advantage of employing temporal information to analyze dynamic multi-mode networks. Shaw and Jebara [30] exploited a semidefinite program to learn a low-dimensional representation that well preserves the global topological structure. Mei et al. [25] designed a harmonic regularization based embedding framework to tackle the problem of topic modeling with network structure. Ahmed et al. [2] proposed an asynchronous distributed matrix factorization algorithm for large-scale graphs. Bourigault et al. [5] projected the observed temporal dynamic into a latent space to better model the information diffusion in networks. Grover and Leskovec [11] further advanced the random walk based embedding algorithms by adding flexibility in exploiting neighborhoods. To embed heterogeneous networks, Jacob et al. [14] extended the transductive models and deep learning techniques into the problem. Yang et al. [45] exploited a probabilistic model to conduct network embedding in a semi-supervised manner. Most recently, several deep learning based embedding algorithms [27, 42, 43] were proposed to further enhance the performance of learned representations.

Attributed network analysis is put forward due to the fact that numerous networks are often associated with abundant content describing attributes of each node. In these networks, it has been widely accepted that there exist correlations among geometrical structure and node attributes [22, 46]. Therefore, algorithms [10, 12, 34] exploiting them together could improve the overall learning performance. For instance, Tsur and Rappoport [40] advanced the prediction of spread of ideas by analyzing both social graph topology and content. In order to tackle the complex data structures, several efforts [28, 47] have been devoted to jointly embedding the two information sources into a unified space. Qi et al. [28] explored an effective algorithm that jointly embeds context and content in social media by constructing a latent space of semantic concepts. Le and Lauw [18] advocated a holistic framework for handling both document linkage and textual information and finding a unified low-dimensional representation. They achieved this via a joint probabilistic model. Li et al. [19] exploited the possibility of jointly learning latent factors in high-dimensional content data and link information via a streaming feature selection framework. Chang et al. [7] transformed content into another network and exploited a nonlinear multi-layered embedding model to learn the complex interactions between the constructed content network and original network.

In many applications, data exhibits multiple facets of presentations, and these data are referred as multi-view data. Multi-view learning [17, 20] aims at learning a statistical model from multiple information sources. A number of algorithms have been proposed in the literature. Qian et al. [29] investigated a reconstruction error based framework for handling multi-label and multi-view learning, which can explicitly quantify the performance of multiple labels or views merging. Lou et al. [20] applied a two-side multimodal neural network to embed words based on multiple data sources. A more detailed review of multi-view learning can be referred to [32]. The main differences between our work and multi-view learning are the facts that an attributed network can be seen as one specially constructed data source, and ANE itself is a challenging problem. Labels are also a special category of source with particular modality.

6. CONCLUSIONS AND FUTURE WORK

Label data is a distinct and essential information source observed in a variety of attributed networks, which is beneficial to ANE. Incorporating labels into embedding representation learning is promising but challenging. To this end, we propose a novel framework LANE, which jointly projects an attributed network and labels into a unified embedding space by extracting their correlations. Specifically, we first uniformly model the structural proximities in the attributed network and labels based on pairwise similarities, and then jointly map them into an identical embedding space via three relevant correlation projections. Extensive experiments on BlogCatalog and Flickr datasets demonstrate that LANE consistently performs the most effective embedding representation. Our results lead to the following open questions that we plan to study in future work: (1) Some label informed networks are evolving, so how can we extend the proposed method to embed this type of networks dynamically? (2) Sometimes we may only be able to collect partial node attributes or label data. How can we design robust models to tackle these situations?

7. REFERENCES

[1] L. A. Adamic and B. A. Huberman. Zipf's law and the internet. *Glottometrics*, 2002.

[2] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola. Distributed large-scale natural graph factorization. *WWW*, 2013.

[3] F. R. Bach and M. I. Jordan. Learning spectral clustering. *NIPS*, 2004.

[4] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *NIPS*, 2001.

[5] S. Bourigault, C. Lagnier, S. Lamprier, L. Denoyer, and P. Gallinari. Learning social network embeddings for predicting information diffusion. *WSDM*, 2014.

[6] L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *U.S.S.R. Computational Math. and Math. Phys.*, 1967.

[7] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang. Heterogeneous network embedding via deep architectures. *KDD*, 2015.

[8] F. R. Chung. Spectral graph theory. *American Mathematical Soc.*, 1997.

[9] I. S. Filotti, G. L. Miller, and J. Reif. On determining the genus of a graph in $O(v^{O(g)})$ steps. *STOC*, 1979.

[10] S. Gao, L. Denoyer, and P. Gallinari. Temporal link prediction by integrating content and structure information. *CIKM*, 2011.

[11] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. *KDD*, 2016.

[12] X. Hu, L. Tang, J. Tang, and H. Liu. Exploiting social relations for sentiment analysis in microblogging. *WSDM*, 2013.

[13] T. Iwata, T. Yamada, and N. Ueda. Probabilistic latent semantic visualization: topic model for visualizing documents. *KDD*, 2008.

[14] Y. Jacob, L. Denoyer, and P. Gallinari. Learning latent representations of nodes for classifying in heterogeneous social networks. *WSDM*, 2014.

[15] I. Joliffe. Principal component analysis. *Springer Verlag*, 1986.

[16] D. B. Kandel. Homophily, selection, and socialization in adolescent friendships. *American Journal of Sociology*, 1978.

[17] A. Kumar, P. Rai, and H. Daume. Co-regularized multi-view spectral clustering. *NIPS*, 2011.

[18] T. M. V. Le and H. W. Lauw. Probabilistic latent document network embedding. *ICDM*, 2014.

[19] J. Li, X. Hu, J. Tang, and H. Liu. Unsupervised streaming feature selection in social media. *CIKM*, 2015.

[20] Y. Luo, J. Tang, J. Yan, C. Xu, and Z. Chen. Pre-trained multi-view word embedding using two-side neural network. *AAAI*, 2014.

[21] P. V. Marsden. Homogeneity in confiding relations. *Social Networks*, 1988.

[22] P. V. Marsden and N. E. Friedkin. Network studies of social influence. *SAGE Focus Editions*, 1994.

[23] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 2001.

[24] A. I. Mees, P. E. Rapp, and L. S. Jennings. Singular-value decomposition and embedding dimension. *Phys. Rev. A*, 1987.

[25] Q. Mei, D. Cai, D. Zhang, and C. Zhai. Topic modeling with network regularization. *WWW*, 2008.

[26] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. *NIPS*, 2002.

[27] B. Perozzi, R. Al-Rfou, and S. Skiena. DeepWalk: Online Learning of Social Representations. *KDD*, 2014.

[28] G. Qi, C. Aggarwal, Q. Tian, H. Ji, and T. Huang. Exploring context and content links in social media: A latent space method. *TPAMI*, 2012.

[29] B. Qian, X. Wang, J. Ye, and I. Davidson. A reconstruction error based framework for multi-label and multi-view learning. *TKDE*, 2015.

[30] B. Shaw and T. Jebara. Structure preserving embedding. *ICML*, 2009.

[31] A. P. Singh and G. J. Gordon. Relational Learning via Collective Matrix Factorization. *KDD*, 2008.

[32] S. Sun. A survey of multi-view machine learning. *Neural Computing and Applications*, 2013.

[33] J. Tang, C. Aggarwal, and H. Liu. Node classification in signed social networks. *SDM*, 2016.

[34] J. Tang, H. Gao, X. Hu, and H. Liu. Exploiting homophily effect for trust prediction. *WSDM*, 2013.

[35] J. Tang, J. Liu, M. Zhang, and Q. Mei. Visualizing large-scale and high-dimensional data. *WWW*, 2016.

[36] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. LINE: Large scale Information Network Embedding. *WWW*, 2015.

[37] L. Tang, H. Liu, J. Zhang, and Z. Nazeri. Community evolution in dynamic multi-mode networks. *KDD*, 2008.

[38] J. Tenenbaum, V. Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 2000.

[39] L. N. Trefethen and D. Bau III. Numerical linear algebra. *SIAM*, 1997.

[40] O. Tsur and A. Rappoport. What's in a hashtag? content based prediction of the spread of ideas in microblogging communities. *WSDM*, 2012.

[41] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 2007.

[42] D. Wang, P. Cui, and W. Zhu. Structural deep network embedding. *KDD*, 2016.

[43] J. Weston, F. Ratle, H. Mobahi, and R. Collobert. Deep learning via semi-supervised embedding. *Neural Networks: Tricks of the Trade*, 2012.

[44] S. Yan, D. Xu, B. Zhang, HJ. Zhang, Q. Yang, and S. Lin. Graph embedding and extensions: A general framework for dimensionality reduction. *PAMI*, 2007.

[45] Z. Yang, W. Cohen, and R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *ICML*, 2016.

[46] Z. Yang, J. Guo, K. Cai, J. Tang, J. Li, L. Zhang, and Z. Su. Understanding retweeting behaviors in social networks. *CIKM*, 2010.

[47] S. Zhu, K. Yu, Y. Chi, and Y. Gong. Combining content and link for classification using matrix factorization. *SIGIR*, 2007.