A United Approach to Learning Sparse Attributed Network Embedding

Hao Wang[1], Enhong Chen[1,*], Qi Liu[1], Tong Xu[1], Dongfang Du[2], Wen Su[2], Xiaopeng Zhang[2]
[1]Anhui Province Key Laboratory of Big Data Analysis and Application, School of Computer Science and Technology,
University of Science and Technology of China
{wanghao3}@mail.ustc.edu.cn, {cheneh,qiliuql,tongxu}@ustc.edu.cn
[2]Tencent Inc, China
{blithedu, oliviayuisu,xpzhang}@tencent.com

***Abstract*—Recently, the Network Representation Learning (NRL) techniques, which target at learning the low-dimension vector representation of graph structures, have attracted wide attention due to the effectiveness on various social-oriented application. Though large efforts have been made on the joint analysis combining node attributes with the network structure, they may usually fail to summarize the weighted correlations within nodes and attributes, especially when the nodes suffer extremely sparse attributes. To that end, in this paper, we propose a novel Sparse Attributed Network Embedding (SANE) framework to learn the network structure and sparse attribute information simultaneously in a united approach. Specifically, we first embed the nodes and attributes into a low-dimensional vector space. Then we introduce the pairwise method to capture the interaction between nodes and sparse attributes, and aggregate the attribute information of neighbors to alleviate sparsity for obtaining a better vector representation of node embeddings, which will be used in following network representation learning task. Along this line, we maintain the network structure by maximizing the probability of predicting the center node according to surrounding context nodes. Different from previous work, we introduce an attention mechanism to adaptively weigh the strength of interactions between each context node and the center node, according to the node attribute similarity. Furthermore, we combine the attention network with CBOW model to learn the similarity of the network structure and node attributes simultaneously. Extensive experiments on public datasets have validated the effectiveness of our SANE model with significant margin compared with the state-of-the-art baselines, which demonstrates the potential of adaptively attribute analysis in network embedding.**

***Index Terms*—network embedding, attributed network, attention mechanism, sparse attributes**

I. Introduction

With the development of embedding techniques, a series of network representation learning (NRL) algorithms, which target at learning the low-dimension vector representation for the network structure, have been proposed to support various social-oriented applications, such as node classification [1], [2], link prediction [3], [4], network clustering [5] and social influence analysis [6], [7]. Traditionally, prior arts focused on the local or global network structure to derive the objective functions with employing first-order and second-order proximity [8], [9]. However, some other information except for the network structure, like node attributes, has been largely ignored in previous research work.

Indeed, for practical scenario in the real world, node attributes, such as user profiles or preferences, could be beneficial for revealing the network structure since they may crucially impact the pairwise connections and interactions with nodes. Correspondingly, some related work attempted to jointly learn the node embedding with integrating attribute information to achieve better performance [10]–[13]. For instance, TADW [10] first utilized inductive matrix completion to incorporate the text information of nodes. LANE [11] combined the graph matrix and attribute similarity matrix to project them into a common vector space, and UPP-SNE [12] generated the node embedding via a non-linear mapping from node attributes. However, though these methods have enhanced the graph embedding performance, unfortunately, they could be disturbed by the sparse and incomplete node attribute information, which is common in the application scenario. Moreover, the connection strength within different nodes should be distinguished, which have usually been neglected in prior arts. Thus, more comprehensive frameworks for attributed network embedding are still urgently required.

To deal with these tasks, in this paper, we propose a novel **S**parse **A**ttributed **N**etwork **E**mbedding (**SANE**) model to learn the network structure and node attribute information in a united approach. In order to model the relationship between nodes and sparse attributes, we first embed the nodes and attributes into a low-dimensional vector representation. Then we utilize a pairwise method to capture the correlation between nodes and attributes. Besides, we aggregate neighbors' attribute information to further alleviate sparsity for obtaining a better representation of node embedding. In order to learn the network structure information, we conduct the truncated random walk to generate training sequences. By maximizing the probability of predicting center node according to context nodes, the network embeddings of nodes with the similar network structure are closer. Different from previous work, we introduce the attention mechanism [14], [15] to assign

[*] denotes the corresponding author.

978-1-5386-9159-5/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDM.2018.00071

the interaction weight between each context node and center node according to the similarity of attributes. Furthermore, we combine the attention mechanism with CBOW [16] model to learn the similarity of network structure and node attributes simultaneously. With this united approach, we eventually make the vector representation of nodes with the similar network structure and attributes closer. Generally, the technical contribution of our paper could be briefly summarized as follows:

- In order to model the relationship between nodes and sparse attributes, we propose a novel pairwise method to capture correlation between nodes and attributes for generating a better representation of node embedding.
- We introduce the attention mechanism to weigh the strength of interactions between nodes, and combine with CBOW model to learn the similarity of the network structure and attributes in a united framework.
- Extensive experiments on public datasets have validated the effectiveness of our SANE model with significant margin compared with the state-of-the-art baselines, which demonstrates the potential of adaptive attribute analysis in network embedding.

The rest of this paper is organized as follows. We summarize the related work in Section II. Preliminaries of our model are described in Section III. Then we propose our SANE model with detailed formulation and optimization in Section IV. In Section V, extensive experiments and discussion will be reported. Finally, we summarize our paper in Section VI.

II. Related work

A. Network Embedding

In recent years, network embedding methods that only utilized the network structure information are the most studied in the field of network representation learning. These approaches can be divided into three categories: The first is based on truncated random walks and assumes that nodes with the similar network structure have similar vector representations. DeepWalk [17] first attempts to generate training samples by random walks on network, and utilizes the skip-gram model proposed in Word2vec [16] to learn the vector representation of nodes. Noticing that DeepWalk uses the uniform sampling to generate the training sentences, node2vec [18] conducts the weighted random walk by two hyperparameters p and q, in order to capture the homogeneity and structure equivalence respectively. The second is based on k-order distance between nodes in network. For example, LINE [8] focuses on preserving first-order proximity and second-order proximity to learn the node representation. Then GraRep [9] further captures k-order relational structure information to enhance node representation by manipulating global transition matrices. The third is based on deep learning techniques. With the advantage of deep learning, we can obtain higher-order nonlinear representation [19], [20]. Therefore, SDNE [21] proposes a semi-supervised auto-encoder model to obtain node embedding by preserving the global and local network structure information. DNGR [22] adopts a random surfing model to capture the graph structural information and learns the node representation from PPMI matrix by utilizing stacked denoising auto-encoder. Recently, some of the latest work introduce generative adversarial network into the network representation learning. GraphGAN [23] proposes an innovative graph representation learning framework that the generator learns the underlying connectivity distribution and the discriminator predicts the probability the edge existence between a pair of vertices. ANE [24] leverages the adversarial learning principle to regularize the node representation. It consists of structure preserving component and adversarial learning component, which aims to capture the network structure properties and match the distribution of node representation to given priors respectively.

These methods introduced above only utilize the network structure information. Furthermore, some researchers attempt to combine the node attribute information to enhance the node representation. TADW [10] first uses inductive matrix complement framework to incorporate the text features into network representation. Based on TADW, HSCA [25] captures the homogeneity of network by adding regularization of adjacent nodes. HNE [26] utilizes the multi-layer neural network to map nodes and different attributes to the same vector space, and learn the node representation by preserving the first-order proximity. CENE [27] further uses the different types of RNN to extract high-order text features. Then UPP-SNE [12] utilizes a nonlinear mapping method to embed social user profile information into a consistent subspace, where the network seamlessly encoded to jointly learn informative node representation. GraphSAGE [28] iteratively generates the node embedding by sampling and aggregating features from the nodes' local neighborhood. Different from these methods, our approach focuses on how to combine the sparse node attributes and jointly learn the similarity of the network structure and attributes in a unified framework.

Furthermore, some research work formalize it into a supervised or semi-supervised problem by incorporating node label information. TriDNR [29] learns node representation by modeling the inter-node relationship, node-word correlation and label-word correspondence simultaneously. LANE [11] proposes to learn the representation of nodes, attributes, labels via spectral techniques respectively, and projects them into a common vector space to obtain the node embedding. M-NMF [30] utilizes a novel Modularized Nonnegative Matrix Factorization to incorporate the community structure into network embedding. Planetoid [31] uses a graph-based semi-supervised learning framework that the node representation is jointly trained to predict the class label and the context in the graph. GCN [32] is based on an efficient variant of convolution neural networks which operate directly on graphs and optimizes the node representation in semi-supervised learning graph framework. However, it's often difficult to obtain the node label information in real-world scenarios. Therefore, we focus on learning the attributed network representation with unsupervised learning in this paper, which could be easily applies to more scenarios. Finally, more related work on network embedding can be found in this survey [33].

TABLE I: Terms and Notations

Notations	Description
$G = G(V, E, A)$	a graph with attribute matrix A
U	node embedding matrix
U'	node context embedding matrix
P	node offset embedding matrix
F	attribute embedding matrix
d	dimension of node embedding
m	dimension of node attributes
b	window size of context nodes

B. Attention Mechanism

Recently, attention mechanism has proved the effectiveness in natural language processing and computer vision research field [14], [15]. The attention mechanism is mainly based on principled of human visual mechanism that human recognition mainly focused on selective parts of the whole perception space. Inspired by this idea, we adopt the soft attention model [14] to assign attentive weights for surrounding context nodes when predicting the center node in the graph. With the help of soft attention mechanism, the more similar attributes of nodes are, the greater weight of nodes will be assigned. Therefore, we can preserve the network structure and make the nodes with similar attributes closer in the vector space by a unified framework. To the best of our knowledge, our proposed SANE model is the first to introduce the attention mechanism in attributed network representation learning.

III. Preliminaries

In this section, we first give the definition of our problem, and review the basic network representation model, DeepWalk. For the better explanation, we summarize key terms and notations in Table I for reference.

A. Problem Definition

In this section, we will introduce the formal expression of the problem. Let $G = (V, E, A)$ denotes an attributed network, where V is the set of nodes, E is the set of edges, $A_{|V| \times m}$ represents the attributes matrix and the i-th row a_i is the attribute vector of the corresponding node i. For example, a_i can represent users' gender, location and profession in social network such as Twitter, Facebook. While in citation networks, a_i can be the words appearing in a paper. And these attributes are often very sparse in the real-world scenarios. Our model takes the network structure and sparse attribute information as input, and outputs the latent vector representation for each node. Further, these node embedding can be directly used in subsequent applications like node classification and link prediction. In the end, we formalize the problem as follows:

Problem 1 (Sparse Attributed Network Embedding): Given a sparse attributed network $G = (V, E, A)$, we aim to learn a low-dimensional vector representation $u_v \in \mathbb{R}^d (d << |V|)$ for each node $v \in V$, such that 1) the representation of nodes with the similar network structure should be more similar; 2) the representation of nodes with similar attributes should be be closer in the latent vector space.

B. DeepWalk

Before introducing our model, we first present the basic network representation learning model, DeepWalk. It's the first to introduce the skip-gram model into network embedding, in order to learn a distributed vector representation for each node to preserve the network structure. Specifically, DeepWalk first takes the truncated random walks from each node to generate the training corpus of node sequences. Based on the assumption that nodes with similar neighborhood are more similar, for the sequence of nodes $S = \{v_1, v_2, ...v_{|S|}\}$, skip-gram model predicts the surrounding nodes $\{v_{i-k}, ..., v_{i+k}\} \setminus \{v_i\}$ according to the center node v_i. Then the node representation is learned by maximizing the likelihood that context nodes are predicted from the center node. Therefore, the objection function of DeepWalk is to maximize the average log probability of the context nodes given by the center node, which is formalized as follows:

$$\mathcal{L} = \frac{1}{|S|} \sum_{i=1}^{|S|} \sum_{-b \leq j \leq b, j \neq 0} \log p(v_{i+j}|v_i), \tag{1}$$

where $p(v_j|v_i)$ is defined by the following softmax function,

$$p(v_j|v_i) = \frac{\exp{(u_j'^T u_i)}}{\sum_{v \in V} \exp(u_v'^T u_i)}, \tag{2}$$

where u_i is the node embedding of v_i when it's treated as center node and u_j' is the node context embedding while it's treated as the 'context node' v_j. DeepWalk finally learns the node embedding by optimizing Equation (1), so that nodes with similar network structures have the similar node embedding. Our model is also based on the same assumption as DeepWalk, but we utilize *the continuous bag-of-words* CBOW [16] model which uses the surrounding context nodes to predict the center node, in order to model the different importance of context nodes. In next section, we will describe it in detail.

IV. Sparse Attributed Network Embedding

In this section, we first present a general description of our model. Then we introduce the model formation and optimization in detail.

In this paper, we propose the **S**parse **A**ttributed **N**etwork **E**mbedding (SANE) model to jointly learn the network structure and sparse attributes, which is illustrated in Figure 1. Similar with DeepWalk, we first take the truncated random walks from each node to generate the training node sequences. In order to characterize the interaction relationship between nodes and sparse attributes, we project them into a low-dimensional and dense vector. Then we propose a pair-wise method to model the correlation between them, and incorporate the neighbors' information to obtain a better representation of node embedding. Finally, we adopt the CBOW [34] model that uses surrounding context nodes to predict the center node, and fuse attention mechanism for assigning different weights according to the similarity of attributes. Therefore, we can preserve the two properties of node embedding defined in Problem 1 in a unified framework, that the representation of nodes with the similar network structure and attributes are closer in the latent vector space. And we will introduce each part of our proposed model in detail.

A. Modeling Nodes and Sparse Attributes

As mentioned in previous section, node attributes in the network are an essential part of learning network representation. In real-world scenarios, node attributes vector a_i are usually high-dimensional and sparse, so it's necessary to model the relationship between nodes and sparse attributes to obtain a better node representation for later embedding learning. To track this issue, we propose a pair-wise method to capture the interaction relationship as follow:

1) Pairwise Node Embedding: First, we project the nodes and attributes to a low-dimensional common vector space and utilize the matrices $\mathbf{P} \in \mathbb{R}^{|v| \times d}$ and $\mathbf{F} \in \mathbb{R}^{|m| \times d}$ to represent the lookup embedding matrix respectively. Then the pairwise node embedding is defined as follows:

$$u_i = \sum_{j=1}^{m} (p_i \odot f_j) \cdot a_{ij}, \tag{3}$$

where p_i denotes the offset node embedding of node i, f_j represents the attribute embedding for attribute j, a_{ij} is the corresponding attribute value, and $\odot$ represents the element-wise product of two vectors. We represent each attribute with a low-dimensional dense vector, so that we can capture and generalize the similarity between attributes. And each node has a unique offset embedding to ensure that nodes with the same attributes can learn different representation in the network. Although the attributes of each node are very sparse and the value of a_i are mostly zero, we can utilize the element-wise operation to effectively capture the interaction relationship between nodes and attributes by the pairwise method.

The method of pairwise node embedding can alleviate the sparsity of node attributes to some degree, but the interaction between nodes and attributes is still very limited. In order to further alleviate this problem, we utilize the neighbors' information of each node to enhance node representation. As we know, there exists the homogeneity phenomenon [35] in the network that adjacent nodes often have similar node attributes. For example, the friends in social network tend to have the similar profiles like age, education and profession, etc. In citation network, the referenced articles often have the similar research directions. Therefore, we take advantage of this property and propose the neighbor-based node embedding, which is defined as follows:

2) Incorporating Neighborhood Information: For each node v_i, we first define the representation of neighbors' attribute information as shown below:

$$f_j^{N(i)} = Pooling(\{f_j \cdot a_{kj} |\ \forall k \in N(i)\}), \tag{4}$$

where $N(i)$ denotes the neighbor nodes of node v_i, $f_j \cdot a_{kj}$ is the representation of neighbor node k on attribute j. Because the number of nodes' neighbors is inconsistent, we utilize the average pooling function to transform the sets of vectors to obtain a fixed length vector representation. Therefore, $f_j^{N(i)}$ can provide the comprehensive representation of the neighbors $N(i)$ on attribute j. Then we integrate the neighbors' attribute information to obtain the final node embedding as following:

$$u_i = \sigma(\lambda \sum_{j=1}^{m} (p_i \odot f_j) \cdot a_{ij} + (1-\lambda) \sum_{j=1}^{m} (p_i \odot f_j^{N(i)})), \tag{5}$$

where λ is the hyper-parameter to balance the relationship between nodes and surrounding attributes in different networks, and the final node embedding is composed of two parts. Although the nodes' own attributes are very sparse, we can obtain more attribute information from the neighbors. Besides, we utilize sigmoid function $\sigma(x)$ to make non-linear transformation of final node embedding for learning a more robust vector representation.

B. Preserving Network Structure

In previous section, we introduce how to utilize attribute information to obtain a better node representation. And in this section, we will describe how to learn the similarity of node attributes and the network structure in a unified framework. We first take the truncated random walks from each node to generate the training corpus. Based on the assumption that representation of nodes with similar neighborhood are similar, we adopt the CBOW model that uses the surrounding context nodes to predict the center node and define the objective function as follow:

1) Objective Function: For each node v_i in training sentences, we utilize the surrounding context nodes $\{v_{i-k}, ..., v_{i+k}\} \setminus \{v_i\}$ to predict the center node, and maximize the log probability to preserve the network structure, which is formulated as follows:

$$\begin{aligned} \mathcal{L} &= -\log p(v_i | context(v_i)) \\ &= -\log \frac{\exp u_i'^T \, u_{context(i)}}{\sum_{j=1}^{|V|} \exp u_j'^T \, u_{context(i)}}, \end{aligned} \tag{6}$$

where u_i' is node context embedding, $u_{context(i)}$ is the comprehensive representation of the surrounding context nodes. Different from previous work, we utilize the CBOW model instead of skip-gram model to characterize the different importance of context nodes. Because the nodes having the similar attributes with the center node should be more important when predicting. Therefore, we assign different weights to each context node, and the vector representation of surrounding context nodes is calculated as follows:

$$u_{context(v_i)} = \sum_{j \in [i-b, i+b] \setminus \{0\}} s_{ij} u_j, \tag{7}$$

where u_j is the node embedding defined in previous section, b is the window size of surrounding context nodes, and s_{ij} is the weight value of node v_j, which is calculated by the following attention network.

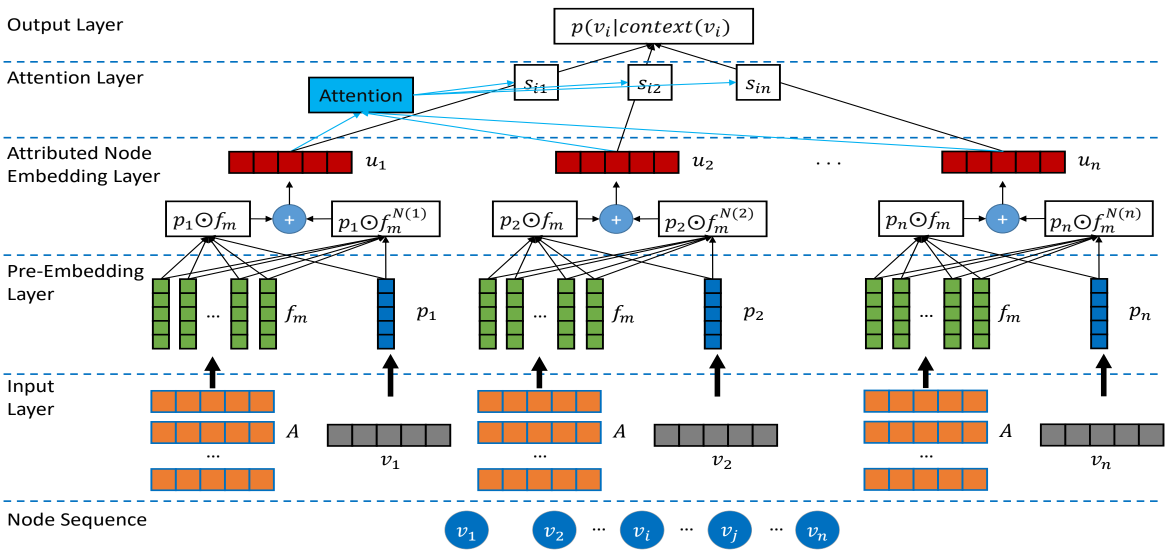


Fig. 1: Framework of Sparse Attributed Network Embedding Model(SANE).

2) Attention Network: We utilize the attention network to assign weight s_{ij} to each context node v_j, so that the greater weight of context node v_j corresponds, the more similar vector representations of center node v_i and context node v_j are. Formally, the attention network is defined as:

$$s'_{ij} = \mathbf{h}^T ReLU(\mathbf{W}[u_j, u'_i]), \tag{8}$$

where u'_i is the context embedding of the center node v_i, $\mathbf{W} \in \mathbb{R}^{t \times 2d}$ and $\mathbf{h} \in \mathbb{R}^t$ are model parameters, $[u_j, u'_i]$ represents the combination of vectors and $ReLU(x) = max(0, x)$. Furthermore, the attention scores are normalized by the softmax function:

$$s_{ij} = \frac{\exp(s'_{ij})}{\sum_{(i,j) \in R_x} \exp(s'_{ij})}. \tag{9}$$

Because the defined node embedding incorporates attribute information, the nodes with the similar attributes will be assigned a greater weight by attention network, so that the vector representations of nodes are closer. Furthermore, we combine the attention mechanism with CBOW model. By optimizing the final objective function Equation (6), we eventually enable node embedding to learn the similarity of attributes and the network structure simultaneously in a unified framework.

C. Model Learning

In this section, we will introduce the model optimization and parameter initialization.

1) Negative Sampling: By optimizing the objective function, we can eventually learn the node representation. However, directly optimizing Equation (6) is computationally expensive, because the denominator of $p(v_i|context(v_i))$ requires summation over all nodes in the network, where the number of nodes is usually very large. To address this problem, we adopt the approach of negative sampling proposed by [16], which selects negative samples according to the noisy distribution $P(v)$ for each node context. Therefore, the objective function Equation (6) is finally replaced by the following:

$$\log \sigma(u_i'^T\, u_{context(i)}) + \sum_{t=1}^{neg} E_{v_t \sim P(v)}[\log \sigma(-u_t'^T\, u_{context(i)})], \tag{10}$$

where $\sigma(x)$ is sigmoid function, and neg is the number of negative samples. We set the node noisy distribution $P(v) \propto d_v^{3/4}$ as proposed in [16], where d_v is out-degree of node v.

2) Parameter Initialization: As we know, parameter initialization has a great impact on the convergence and results of the model. So we introduce the important parameters of our model in detail in this section. We initialize the node context embedding to $\vec{\mathbf{0}}$ vectors and attribute embedding with a Gaussian distribution (with a mean of 0 and standard deviation of 0.01). Specially for node offset embedding, we initialize it all to $\vec{\mathbf{1}}$ vector and explain the reason.

The method of parametric embedding [36], [37] is widely utilized to integrate the attribute information into representation vector, and has proved the effectiveness in many other research fields [37], [38], which is defined as follows:

$$u_i = M^T a_i, \tag{11}$$

where $M \in \mathbb{R}^{m \times d}$ is the transformation matrix, and a_i is the attribute vector of node v_i. Furthermore, we discuss the relationship between the parametric embedding and our proposed pairwise attributed node embedding. If we regard each row M_j of matrix M as the vector representation f_j of attribute j, Equation (11) can be formulated as :

$$u_i = \sum_{j=1}^{m} a_{ij} M_j, \tag{12}$$

where m is the dimension of node attributes. We can see that parametric embedding method summarizes the corresponding attribute embedding as the node representation, which doesn't consider the interaction relationship between nodes and attributes. If we set all the node offset embedding p_i as $\vec{1}$ vector, the pairwise node embedding defined in Equation (3) can be reduced to the same expression as Equation (12). However, we can observe the method of parametric embedding always obtains the identical vector representation of nodes with the same attributes, even if their network locations are different. This demonstrates that the traditional parametric embedding has some limitations in this problem, but our proposed pairwise node embedding is more flexible which can simultaneously learn the network structure and attribute information via node offset and attribute embedding. Therefore, we finally initialize all the node offset embedding as $\vec{1}$ to obtain a better initialization point.

V. Experiments

A. Experimental Settings

1) Datasets: We conduct the experiments on four public datasets, which are two different types of networks that represent social networks and citation networks. For each network, we regard the links between nodes as undirected and remove the nodes that are not connected in the network. The detailed description of the datasets is listed as follows and their statistics are summarized in Table II.

- **Rochester**[1] is the Facebook network constructed by [39], which contain students from University of Rochester. There are $4,563$ nodes and $161,404$ friendship links. Each node's profile is described by seven anonymized attributes: student/faculty status, gender, major, second major/minor, dorm/house, high school and class year. And we encode these attributes as a 241-dimensional binary feature vector.
- **Cora**[2] contains $2,708$ machine learning papers and $5,429$ citation links. These papers are divided into seven categories. Each paper is represented as a binary vector of $1,433$ dimensions indicating the presence of the corresponding word. We regard these binary vectors as the node attributes.
- **Citeseer** contains $3,312$ publications of six classes. There are $4,732$ citation links in the network and each document is represented as a binary vector of $3,703$ dimensions, which is treated as the node attributes like Cora.
- **DBLP**[3] contains $60,744$ papers and $52,890$ citation links, which is constructed by [29]. They extract the papers from four research areas, i.e. *database*, *data mining*, *artificial intelligent* and *computer vision*, which can be utilized as the ground-truth label of each node. And each article title is used as the node attribute, represented by a $3,799$ dimensional TF-IDF vector. We filter out the articles without any citation relationship and retain the network of $17,725$ nodes and $105,781$ links.

TABLE II: Statistics of the datasets

Datasets	**Nodes#**	**Links#**	**Features#**	**Sparsity#**
Rochester	$4,563$	$161,404$	241	2.9%
Cora	$2,708$	$5,429$	$1,433$	1.2%
Citeseer	$3,327$	$4,732$	$3,703$	0.8%
DBLP	$60,744$	$52,890$	$3,799$	0.1%

2) Baselines: In order to demonstrate the effectiveness of our proposed SANE model, we compare with the representative state-of-the-art network representation learning algorithms. Since SANE utilizes the network structure and node attribute information, the baselines are selected from two aspects. One is the algorithms only using the network structure information like Node2vec and LINE. The other is methods that utilize both the network structure and node attributes such as TADW, UPP-SNE and GraphSAGE. Because our SANE model is an unsupervised algorithm without incorporating label information, we will not compare with the semi-supervised methods introduced in related work. And the details of these baselines are illustrated as follows:

- **Attri**: The dimensions of node attributes are usually very large. So we first reduce the dimension of node attributes to 200 via Singular Value Decomposition (SVD) proposed in [10]. Then we utilize the reduced dimension vectors as node embedding.
- **Node2vec** [18]: Different from DeepWalk, it designs a biased truncated random walks to efficiently explore diverse neighborhood and utilize the skip-gram model to learn node embedding[4].
- **LINE** [8]: It learns the network embedding by preserving the *first-order proximity* or *second-order proximity* of the network structure separately[5]. And we utilize the best of them as the final baseline.
- **Node2vec+Attri**: We concatenate the vectors from both Attri and Node2vec as the final node embedding.
- **LINE+Attri**: The node vectors of Attri and LINE are concatenated to generate the final node embedding.
- **TADW** [10]: It is a method based on inductive matrix completion that incorporates node attributes into network embedding learning. It utilizes both the network structural and node attribute information[6].
- **UPP-SNE** [26]: It constructs node embedding via a nonlinear mapping from node attributes and preserves the network structure, in order to jointly learn the node embedding. This method demonstrates that it has substantial performance on sparse node attributes.
- **GraphSAGE** [28]: It first generates the node embedding by sampling and aggregating features from the nodes' local neighborhood. Then the node embedding are learned by maintaining the network structure.
- **SANE-S**: It is the simplified version of our model without incorporating neighbors' attribute information and attention network.

[1] https://escience.rpi.edu/data/DA/fb100/

[2] http://linqs.cs.umd.edu/projects/projects/lbc/index.html

[3] http://arnetminer.org/citation (V4 version is used)

[4] https://github.com/aditya-grover/node2vec

[5] https://github.com/tangjianpku/LINE

[6] https://github.com/albertyang33/TADW

TABLE III: Results of multi-label node classification on **Cora** dataset with $\lambda = 0.5$.

Training ratio	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Node2vec	58.48	65.55	67.95	70.12	71.43	72.21	73.06	74.41	75.07	75.25
LINE	41.11	45.96	50.06	52.32	54.46	56.74	57.31	59.62	59.81	60.47
Node2vec + Attri	60.31	67.96	72.26	74.16	74.88	75.44	75.73	76.55	76.98	77.43
LINE + Attri	43.98	53.50	59.07	61.99	64.32	66.04	67.32	68.90	69.48	71.20
TADW	55.72	64.95	70.64	73.53	75.09	76.72	78.07	78.67	79.49	80.40
UPP-SNE	62.78	72.59	75.05	77.66	78.23	78.63	79.38	79.40	80.04	80.27
GraphSAGE	62.39	68.78	73.83	74.87	75.82	76.97	77.68	77.79	78.46	79.59
SANE-S	64.17	74.29	77.97	79.13	80.38	80.49	81.02	81.36	82.04	82.42
SANE-A	70.10	77.14	79.73	80.56	82.33	82.25	83.40	83.49	83.99	84.11
SANE	**70.55**	**78.78**	**80.23**	**82.31**	**82.50**	**82.80**	**83.60**	**83.64**	**84.01**	**84.47**

TABLE IV: Results of multi-label node classification on **Citeseer** dataset with $\lambda = 0.7$.

Training ratio	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Node2vec	40.51	44.05	47.71	49.22	50.48	51.43	52.00	52.54	52.94	53.15
LINE	29.53	34.16	35.74	37.32	38.92	40.33	40.41	41.45	42.5	43.02
Node2vec + Attri	48.02	57.62	60.52	62.27	63.09	64.22	65.16	65.30	66.15	66.36
LINE + Attri	43.28	51.28	56.04	59.26	59.84	61.77	62.46	63.08	63.25	64.07
TADW	46.65	56.38	61.38	64.68	66.87	67.28	67.39	68.65	69.78	70.06
UPP-SNE	55.62	64.40	66.24	66.91	66.98	68.07	68.42	68.44	68.59	68.85
GraphSAGE	60.34	65.87	66.55	68.51	68.93	69.46	69.83	70.15	70.33	70.41
SANE-S	61.27	67.08	68.57	69.59	69.62	70.24	70.71	70.77	70.90	71.06
SANE-A	62.13	68.37	69.80	70.54	70.86	71.03	71.38	71.84	71.94	72.11
SANE	**66.56**	**70.02**	**70.96**	**71.68**	**71.70**	**72.20**	**72.35**	**72.97**	**72.98**	**73.27**

TABLE V: Results of multi-label node classification on **DBLP** dataset with $\lambda = 0.6$.

Training ratio	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Node2vec	73.12	76.61	77.96	78.65	79.06	79.44	79.65	79.78	79.81	79.85
LINE	65.44	69.04	70.96	72.36	73.12	73.26	73.94	74.20	74.46	74.61
Node2vec + Attri	76.14	78.01	78.56	78.92	79.51	79.79	80.04	80.15	80.39	80.50
LINE + Attri	66.10	71.35	73.69	74.89	76.07	76.78	77.14	77.91	78.02	78.33
TADW	72.42	77.07	79.18	80.35	80.44	80.96	81.23	81.57	81.69	81.79
UPP-SNE	77.13	78.17	78.42	79.30	79.59	79.98	80.05	80.18	80.52	80.62
GraphSAGE	77.85	78.08	79.68	80.75	81.21	81.56	81.87	82.02	82.21	82.27
SANE-S	77.64	78.48	79.35	79.80	79.93	80.31	80.34	80.43	80.58	80.67
SANE-A	78.48	80.02	80.44	80.76	81.15	81.30	81.38	81.41	81.48	81.60
SANE	**80.87**	**81.94**	**82.46**	**82.84**	**82.95**	**83.01**	**83.19**	**83.21**	**83.22**	**83.39**

- **SANE-A**: It is the reduced version of our model only without introducing attention network.

3) Evaluation Metrics: In our experiments, we perform the tasks of multi-label classification, link prediction and visualization to validate different methods. For the multi-label classification task, we adopt the micro-f1 to evaluate the performance as many previous work [17], [18]. We also conduct experiments with macro-f1 and accuracy as the evaluation metrics and obtain the similar trend, so we omit it for brevity. For the link prediction task, we exploit the widely used top-n ranking metrics: Precision, Recall and F1. We get the same experimental results on these metrics. Finally, we only present the results of Precision@K and Recall@K in order to save space, and we set K=5 as it is useless to predict too many nodes in the real world.

4) Parameter Settings: We implement our method based on Tensorflow[7]. We randomly initialize model parameters with a Gaussian distribution (with a mean of 0 and standard deviation of 0.01). For the generation process of training sequences, we set the window size as 5, walk length as 20 and walks per node as 20. And we set the number of negative samples as 5 and mini-batch size as 64 during the stochastic gradient descent optimization. The hyper-parameter λ is tuned by using grid search on the validation set. The parameters of other baselines are the same as those in their original paper and tuned to be optimal. Besides, we set the embedding dimension to be 100 for all methods in order to get a fair comparison.

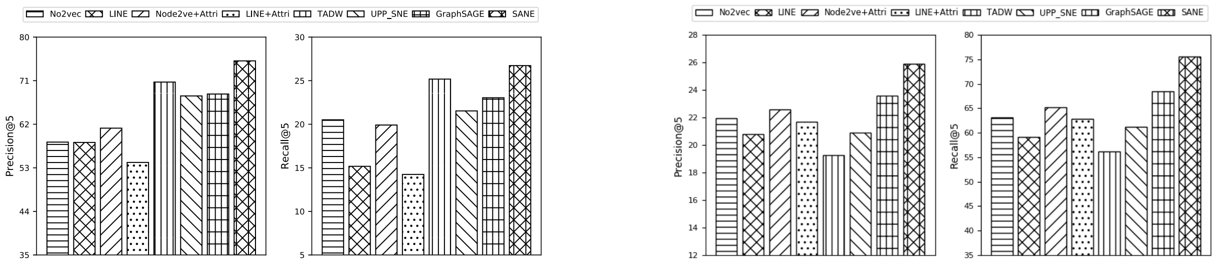
B. Experimental Results

In this section, we first evaluate the performance on multi-label node classification. Then we report the result of link prediction and display the visualization of node embedding.

1) Multi-label Node Classification: For network embedding representation learning, multi-label node classification is a very important task to evaluate the effectiveness of node embedding [40]. The node representations are generated from the network embedding methods and are utilized as node features to classify each node into a set of labels. The same as previous work [18], [29], we select the linear SVM implemented by LibLinear [41] as the classifier, in order to reduce the impact of classifiers on the classification performance.

We randomly select a portion of the labeled nodes as the training data and the rest as test. Besides, we vary the training ratio from 1% to 10% by an increment of 1%. For each

[7] https://github.com/tensorflow/tensorflow



(a) Rochester

(b) Cora

Fig. 2: Performance of link prediction on Rochester and Cora datasets

training ratio, we repeat the experiments for 10 times and report the average results. The comparison results on Cora, Citeseer, DBLP datasets are shown in Table III, Table IV and Table V, respectively. And we utilize **bold-faced** to highlight the best experimental results. From these tables, we can have following observations:

First, compared with all the baselines, our simplified and complete models consistently achieve significant improvement on all datasets with different training ratios. Especially, in the small training proportion, our models can achieve larger improvements. It demonstrates that the learned node embeddings of our models are more effective and robust than baselines' with respect to the node classification task.

Second, compared with Attri, Node2vec and LINE methods, our models achieve better performance. It demonstrates that only using node attributes or network structure is not enough to learn a better node embedding. Although Node2vec+Attri and LINE+Attri methods use both information, their performances are not satisfactory. Because these methods just concatenate two parts of the information, and do not effectively integrate them together to learn node embedding.

Third, TADW, UPP-SNE and GraphSAGE are relatively better methods in baselines, because these methods fuse both the network structure and node attribute information to learn the node representation. From the experimental results, we can observe that TADW exhibits unstable performance under various training ratios and performs poorly under small training samples. Because the target matrix decomposed by TADW is often very sparse and the interaction between nodes and attributes is not considered. For the UPP-SNE method, it doesn't achieve a satisfactory improvement. Although UPP-SNE learns node embedding via non-linear mapping from node attributes, it doesn't effectively alleviate the sparsity of node attributes and strengthen their similarity in the learning process. GraphSAGE is a strong baseline, however, it doesn't consider the attribute sparsity and generate the node embedding by randomly sample K neighbors, which limits the experimental performance.

Finally, from the comparison results of SANE-S and SANE-A, we can illustrate that the addition of neighbors' attribute information can alleviate the sparsity of node attributes. Compared with SANE-A, SANE achieves better performance. It demonstrates that the attention network can adaptively weigh the strength of interactions between nodes and makes the nodes with similar attributes closer.

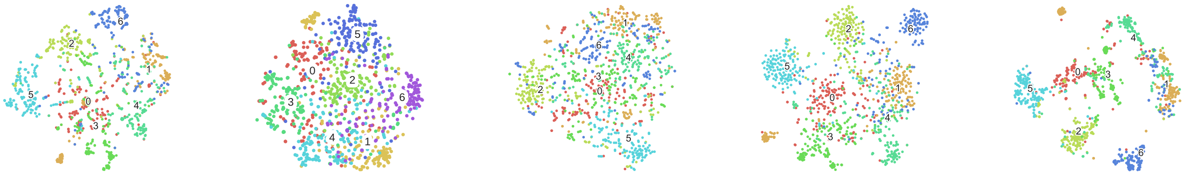
2) Link Prediction: In this section, we conduct the link prediction experiments. Through these experiments, we can demonstrate whether different network embedding algorithms can effectively learn the similarities between nodes in the attributed network.

We use the Rochester and Cora datasets in this section. To conduct the link prediction task, we randomly remove the 30 percentage of edges as the test set and use the rest of the network to train the network embedding model. After training the model, we obtain the vector representation for each node and rank the potential linked nodes based on the cosine similarity. As the number of nodes is often very large, it is impractical to take all nodes as candidates. Therefore, we adopt a similar approach that has been accepted by many previous work [42], [43]: for each node, we randomly sample 100 negative linked nodes that are not connected to it. Then we mix positively linked nodes and the sampled negative nodes together to select the top potential linked nodes of each nodes. We repeat the experiments for 10 times and report the average result. The final results are shown in Figure 2. From these figures, we can obtain the following conclusions:

First, compared with all the baselines, our models outperform the other baselines by a large margin. It proves that the node embeddings learned by our models can better measure the similarity between nodes in the attributed network.

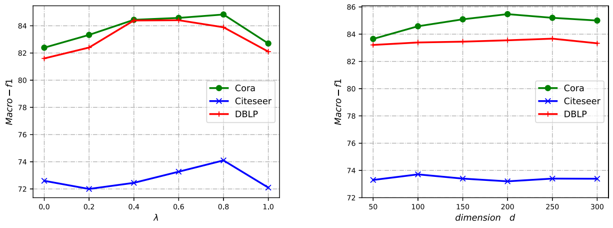
Second, note that the performance of TADW and UPP-SNE is not consistent on different datasets. Their performance on the Cora dataset is even worse than the Node2vec+Attri and LINE+Attri methods. The reason for the poor performance of TADW is that it doesn't provide a clear objective function to preserve the similarity of the network structure. Although the UPP-SNE method achieves the better performance than TADW, its performance still has a large margin with our models, because UPP-SNE can't capture the similarity in node attributes. And the experimental result of GraphSAGE is relatively stable on different datasets. However, it doesn't consider the weight relationship between nodes, which is more important for the link prediction tasks.

Third, from comparison results, we can demonstrate that our proposed SANE model can effectively learn similar relationships between nodes in the network. The best performance



(a) Node2vec (b) TADW (c) UPP-SNE (d) GraphSAGE (e) SANE

Fig. 3: Visualization of network representations learned by different algorithms on the Cora dataset. Each point indicates one node and each color represents one category. The text marked in figures corresponds to the label name.



(a) hyper-parameter λ (b) dimension size d

Fig. 4: Parameter Sensitivity w.r.t the hyper-parameter λ and dimension of node embedding d.

of SANE illustrates that the attention network can effectively reflect the weight relationship between nodes, and can simultaneously learn the similarity of the network structure and node attribute in a united approach.

3) Embedding Visualization: An important application of network embedding is to generate visualizations of a network in a two-dimensional space, so that we can intuitively observe the relevance between nodes in the network. To conduce the visualization task, we randomly select 150 nodes from each category in the Cora dataset. Then we project the node representation learned by different network embedding algorithms into a two-dimensional space by the widely used visualization tool t-SNE [40]. We let nodes with the same label have the same color and mark the label name at the center point of each labels. The visualization result is shown in Figure 3.

From Figure 3, we can observe that the results of Node2vec and TADW are not very meaningful because the nodes with different labels are mixed together. And the visualizations obtained by UPP-SNE and GraphSAGE are better than Node2vec and TADW, in which the nodes from the same label are clustered together. These results indicate that node attributes are important and can be used to alleviate the network sparsity. By comparing with UPP-SNE and GraphSAGE methods, we can observe that our model SANE tends to have a cluster structure and the nodes with same label are closer to each other. Meanwhile, there is an obvious distance between different labels of nodes. These results demonstrate the node representations learned by the SANE model are effective and discriminative, which can also explain why it has better performance on both multi-label classification and link prediction tasks.

C. Parameter Sensitivity

We investigate the sensitivity of our model parameter in this section. Specifically, we mainly evaluate how the hyper-parameter λ and dimension size d affect the performance. We conduct this experiment on three datasets and report the classification performance when the training ratio is 10%.

Impact of the hyper-parameter λ: The parameter λ is utilized to represent the weight of interactions between node and neighbors' attributes. From Figure 4(a), we can observe that when $\lambda = 0$, the performance of our model is the worst. It demonstrates that it's not enough to only consider the interactions between nodes and its attributes. And the optimal value of parameter λ is not consistent on three datasets, because different networks have different characteristics.

Impact of the dimension size d: From the Figure 4(b), we can see the performance raises when the number d of dimension increases. This is because more dimensions are able to learn more useful information. However, the performance decreases when the dimension number d continuously increases. The reason is that too large number of dimensions may introduce more spareness and noises which will reduce the performance.

D. Convergence Analysis

We also conduct experiments to investigate the convergence property of our model. The values of the objective function Equation (6) from iteration 1 to iteration 50 on four networks are recorded. From the record, we can observe that our model can converge to a stable value in **20** iterations on all datasets. Due to space limitations, we omitted the figures here.

VI. Conclusion

In this paper, we proposed a novel **S**parse **A**ttributed **N**etwork **E**mbedding (SANE) framework, which incorporates sparse attributes with the network structure to jointly learn the vector representation for each node. Specifically, we proposed a novel pairwise method to obtain a better representation of node embedding. Then we combined the attention network with CBOW model to simultaneously learn the similarity of the network structure and attributes in a united approach. Compared with the state-of-the-art baselines, the performance of our model has achieved significant improvement. In the future, we will try to utilize the deeper network structure to learn the higher-order nonlinear node representation of the attributed network.

Acknowledgment

This research was partially supported by grants from the National Natural Science Foundation of China (Grants No. U1605251, 61703386 and 91546103), the Anhui Provincial Natural Science Foundation (Grant No. 1708085QF140), and the Young Elite Scientist Sponsorship Program by CAST. Hao Wang and Qi Liu also thank the support of Tencent Rhino-Bird Joint Research Program.

References

[1] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, p. 93, 2008.

[2] S. Bhagat, G. Cormode, and S. Muthukrishnan, "Node classification in social networks," in *Social Network Data Analytics*, 2011, pp. 115–148.

[3] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: Statistical Mechanics and its Applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

[4] D. Du, H. Wang, T. Xu, Y. Lu, Q. Liu, and E. Chen, "Solving link-oriented tasks in signed network via an embedding approach," in *Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics*, 2017, pp. 75–80.

[5] S. N. Soffer and A. Vazquez, "Network clustering coefficient without degree-correlation biases," *Physical Review E*, vol. 71, no. 5, p. 057101, 2005.

[6] Q. Liu, B. Xiang, N. J. Yuan, E. Chen, H. Xiong, Y. Zheng, and Y. Yang, "An influence propagation view of pagerank," *ACM Transactions on Knowledge Discovery from Data*, vol. 11, no. 3, p. 30, 2017.

[7] B. Chang, T. Xu, Q. Liu, and E.-H. Chen, "Study on information diffusion analysis in social networks and its applications," *International Journal of Automation and Computing*, pp. 1–26, 2018.

[8] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 1067–1077.

[9] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015, pp. 891–900.

[10] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Chang, "Network representation learning with rich text information," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015, pp. 2111–2117.

[11] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 2017, pp. 731–739.

[12] D. Zhanga, J. Yinb, X. Zhuc, and C. Zhanga, "User profile preserving social network embedding," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017, pp. 3378–3384.

[13] X. Huang, J. Li, and X. Hu, "Accelerated attributed network embedding," in *Proceedings of the 2017 SIAM International Conference on Data Mining*, 2017, pp. 633–641.

[14] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proceedings of the 32nd International Conference on Machine Learning*, 2015, pp. 2048–2057.

[15] C. Qin, H. Zhu, T. Xu, C. Zhu, L. Jiang, E. Chen, and H. Xiong, "Enhancing person-job fit for talent recruitment: An ability-aware neural network approach," in *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2018, pp. 25–34.

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[17] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.

[18] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 855–864.

[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[20] H. Chen, K. Xiao, J. Sun, and S. Wu, "A double-layer neural network framework for high-frequency forecasting," *ACM Transactions on Management Information Systems*, vol. 7, no. 4, p. 11, 2017.

[21] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1225–1234.

[22] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016, pp. 1145–1152.

[23] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "Graphgan: Graph representation learning with generative adversarial nets," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[24] Q. Dai, Q. Li, J. Tang, and D. Wang, "Adversarial network embedding," *arXiv preprint arXiv:1711.07838*, 2017.

[25] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Homophily, structure, and content augmented network representation learning," in *Proceedings of IEEE 16th International Conference on Data Mining*, 2016, pp. 609–618.

[26] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 119–128.

[27] X. Sun, J. Guo, X. Ding, and T. Liu, "A general framework for content-enhanced network representation learning," *arXiv preprint arXiv:1610.02906*, 2016.

[28] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1025–1035.

[29] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-party deep network representation," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016, pp. 1895–1901.

[30] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 203–209.

[31] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," *arXiv preprint arXiv:1603.08861*, 2016.

[32] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[33] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *arXiv preprint arXiv:1711.08752*, 2017.

[34] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.

[35] M. Weiner, *Japan's minorities: the illusion of homogeneity*, 2009, vol. 38.

[36] L. Maaten, "Learning a parametric embedding by preserving local structure," in *Artificial Intelligence and Statistics*, 2009, pp. 384–391.

[37] T. Iwata, K. Saito, N. Ueda, S. Stromsten, T. L. Griffiths, and J. B. Tenenbaum, "Parametric embedding for class visualization," in *Advances in Neural Information Processing Systems*, 2005, pp. 617–624.

[38] M. A. Carreira-Perpinán and M. Vladymyrov, "A fast, universal algorithm to learn parametric nonlinear embeddings," in *Advances in Neural Information Processing Systems*, 2015, pp. 253–261.

[39] A. L. Traud, P. J. Mucha, and M. A. Porter, "Social structure of facebook networks," *Physica A: Statistical Mechanics and its Applications*, vol. 391, no. 16, pp. 4165–4180, 2012.

[40] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[41] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, no. 8, pp. 1871–1874, 2008.

[42] S.-H. Yang, B. Long, A. Smola, N. Sadagopan, Z. Zheng, and H. Zha, "Like like alike: joint friendship and interest propagation in social networks," in *Proceedings of the 20th International Conference on World Wide Web*, 2011, pp. 537–546.

[43] L. Wu, Y. Ge, Q. Liu, E. Chen, R. Hong, J. Du, and M. Wang, "Modeling the evolution of users preferences and social links in social networking services," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 6, pp. 1240–1253, 2017.