Deep Attributed Network Embedding by Preserving Structure and Attribute Information

Richang Hong, *Member, IEEE*, Yuan He, Le Wu, Yong Ge, and Xindong Wu, *Fellow, IEEE*

***Abstract*—Network embedding aims to learn distributed vector representations of nodes in a network. The problem of network embedding is fundamentally important. It plays crucial roles in many applications, such as node classification, link prediction, and so on. As the real-world networks are often sparse with few observed links, many recent works have utilized the local and global network structure proximity with shallow models for better network embedding. In reality, each node is usually associated with rich attributes. Some attributed network embedding models leveraged the node attributes in these shallow network embedding models to alleviate the data sparsity issue. Nevertheless, the underlying structure of the network is complex. What is more, the connection between the network structure and node attributes is also hidden. Thus, these previous shallow models fail to capture the nonlinear deep information embedded in the attributed network, resulting in the suboptimal embedding results. In this paper, we propose a deep attributed network embedding framework to capture the complex structure and attribute information. Specifically, we first adopt a personalized random walk-based model to capture the interaction between network structure and node attributes from various degrees of proximity. After that, we construct an enhanced matrix representation of the attributed network by summarizing the various degrees of proximity. Then, we design a deep neural network to exploit the nonlinear complex information in the enhanced matrix for network embedding. Thus, the proposed framework could capture the complex attributed network structure by preserving both the various degrees of network structure and node attributes in a unified framework. Finally, empirical experiments show the effectiveness of our proposed framework on a variety of network embedding-based tasks.**

***Index Terms*—Attribute proximity, attributed network embedding, high-order proximity.**

Manuscript received July 8, 2018; revised November 5, 2018; accepted January 18, 2019. Date of publication March 1, 2019; date of current version February 17, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB1002203, in part by the National Natural Science Foundation of China under Grant 61602147, Grant 61602234, Grant 61572032, and Grant 61722204, in part by the Anhui Provincial Natural Science Foundation under Grant 1708085QF155, and in part by the Fundamental Research Funds for the Central Universities under Grant JZ2018HGTB0230. This paper was recommended by Associate Editor E. Chen. *(Corresponding author: Le Wu.)*

R. Hong, Y. He, and L. Wu are with the School of Computer and Information, Hefei University of Technology, Hefei 230009, China (e-mail: hongrc.hfut@gmail.com; heyuan_95@163.com; lewu.ustc@gmail.com).

Y. Ge is with the Department of Management Information Systems, University of Arizona, Tucson, AZ 85721 USA (e-mail: yongge@email.arizona.edu).

X. Wu is with the School of Computing and Informatics, University of Louisiana at Lafayette, Lafayette, LA 70504 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2019.2897152

I. Introduction

In many real-world applications, the data exhibits the information network structure, e.g., the social networks, citation networks, and Internet. In these networks, the nodes represent the entities in the information network, and the edges denote the relationships between entities. Researchers have well recognized that the network data is often sophisticated with sparse edges, and it is technically challenging to directly deal with it. To address this problem, one of the effective strategies is to embed information networks into a low-dimensional space, in which the useful information conveyed by the network is encoded [15]. Therefore, the network embedding has become a hot topic in related areas, such as machine learning and data mining. In these areas, research works on network embedding focus on finding a way to encode graph structure, so that it can be easily exploited by machine learning models. In this paper, we focus on network embedding in machine learning area. As a result, every vertex is represented with a low-dimensional dense vector. The learned low-dimensional embedding is useful in numerous applications, such as node classification [57], link prediction [8], [13], [50], and network visualization [42].

By representing the network as a graph structure, the earlier works on network embedding could be traced back to the classical models, such as IsoMAP [44] and Laplacian Eigenmaps (LE) [3]. These models utilized the graph theory to capture the local proximity between nodes with the observed links. However, the real-world network is very sparse, with many potential links missing and unobserved. By trivially modeling the first-order observed proximity within nodes, the global structure among nodes is neglected in the embedding process. Thus, the performance is unsatisfactory due to the insufficiency of modeling structure preserving properties in the graph. Recently, some researchers argued that, besides the first-order local proximity, it is also important to consider the global structure of the network. For example, the recent works of LINE [43] and SDNE [48] jointly considered the first-order proximity and second-order proximity between vertices in the network. Perozzi *et al.* [35] proposed a DeepWalk algorithm that employed the generated random walk node sequences in a network to capture the second-order and higher-order proximity between nodes. These works advanced the previous works by making full use of the local and global structure of networks, usually leading to better network embedding performance.

In real-world networks, besides the observed node-to-node structure information, many nodes in the network are

associated with rich attribute information. These networks are called the attributed networks. In attributed networks, the network embedding task is more challenging as it involves learning low-dimensional representations of nodes that preserve both the structure and the attribute information. In fact, compared to the network structure embedding, the rich attributes pose both opportunities and challenges for attributed network embedding. On the one hand, researchers have long demonstrated that node attributes are valuable for many network-based applications, such as node classification [57], link prediction [36], community profiling [49], and so on. The rich node attributes provide valuable data sources to alleviate the data sparsity issue in network embedding. On the other hand, nevertheless, it is nontrivial to combine the attribute information in the network embedding process. Several models attempted to tackle this problem by leveraging the attribute information in these previous network embedding-based models [28], [34], [51]. For example, researchers proposed extensions on DeepWalk [35] to incorporate text features of vertices into network representation learning [34], [51]. LANE [28] learned label informed attributed network embedding under the classical matrix factorization-based network embedding models. In summary, these works were based on the previously proposed shallow network embedding models, and alleviated the data sparsity issue by preserving the attribute information in the modeling process. Nevertheless, as the underlying patterns of the attributed network are complex and highly nonlinear, the limited representation ability of shallow models could hardly capture the complex patterns in the attributed networks.

To this end, in order to capture the underlying complex patterns of the attributed network, we propose to devise a deep learning-based framework for learning node representations in attributed networks. This is motivated by the recent success of deep learning-based models for representation learning [40]. By using a cascade of multiple layers of nonlinear units in a neural architecture, these deep learning-based models showed substantial success for automatically learning useful representations from images [22], audio data [31], texts [47], and so on. Thus, researchers from both industry and academia have been in a race to apply deep learning-based methods for a wide range of applications, such as network embedding [12] and recommender systems [18], [19]. However, it is nontrivial to directly apply these deep models for attributed network embedding due to the uniqueness of the attributed network. In order to capture the complex patterns in attributed networks, we need to model the heterogeneity in attributed networks, including preserving the local and global structure of the network, as well as the node attribute information in a designed deep model.

In this paper, we propose a deep attributed network embedding (DANE) that deals with the *data sparsity*, *structure and attribute preserving*, and *nonlinearly* patterns of attributed network embedding in a unified framework. DANE is composed of three steps. First, we adopt a personalized random walk-based model to capture the interaction between network structure and node attributes from various degrees of proximity. In the second step, we construct an enhanced matrix representation of the attributed network by summarizing the various degrees of proximity. With these two steps, the data sparsity issue could be alleviated to some extent with the various degrees of network structure and node attributes. In the meantime, the local and global network structure, as well as the node attributes are well preserved in the enhanced matrix representation. In the third step, we design a deep neural network to exploit the nonlinear complex patterns in the enhanced matrix for network embedding. Thus, the proposed framework could capture the complex attributed network structure by preserving both the various degrees of network structure and node attributes in a unified framework. Finally, we conduct extensive experimental results on three real-world attributed network data. The empirical experiments clearly show the effectiveness of our proposed framework on a variety of network embedding-based tasks.

We organize this paper as follows. In Section II, we review the related work. In Section III, we briefly describe some preliminaries. We present our proposed framework in Section IV, followed by the experimental results in Section V. Finally, we conclude the whole paper in Section VI.

II. Related Work

In this section, we summarize the related work into three categories: 1) classical network embedding models; 2) attributed network embedding models; and 3) deep neural networks.

A. Network Embedding

Network embedding aims to embed information network into a low-dimensional space, in which every vertex is represented as a low-dimensional vector [4], [9]. Earlier works on network representation learning were related to dimension reduction, such as IsoMAP [44] and LE [3]. Recently, proposed models of LINE [43] and SDNE [48] attempted to preserve the first-order proximity and second-order proximity in networks. Specifically, the first-order proximity denotes the observed links in the graph. The second-order proximity of nodes can be interpreted as that nodes with shared neighbors being like to be similar, which is well supported in social theory and linguistics [11]. In LINE, for each node, it first learned the first-order and the second-order network embeddings separately, and then concatenated these two representations as the final node embedding. SDNE is proposed to simultaneously model the first-order proximity and second-order proximity in a semi-supervised deep autoencoder architecture. To better preserve the global network structure, some algorithms were proposed to capture high-order proximity in networks for better embedding performance [5], [6], [35], [45]. Among them, DeepWalk is one of the most popular approaches [35]. DeepWalk used local information from truncated random walks as word-document sequences, and then a classical SkimGram model from natural language modeling to generate vertex representations. DNGR [6] applied a random surfing model to capture graph structural information and applied the stacked denoising autoencoder to generate low-dimensional vertex representations. HOPE is proposed to capture both the higher-order graph structure proximity and the asymmetric transitivity in directed graphs [32]. Due to the success of

convolutional neural networks for image classification, some recent works attempted to tackle the network embedding problems by defining graph convolutional operations [14], [21]. Besides, node embedding is also considered with hashing techniques [27], evolutional graph embedding [29], and so on.

B. Attributed Network Embedding

Attributed network enhances the classical network structure with the detailed node attribute information, such as user profiles in social networks and text messages associated with each article in citation networks. Many previous works were proposed to exploit both structure and attribute information for enhancing network-based tasks, such as link prediction [2], sentiment analysis [41], and community detection [53]. Recently, attributed network embedding has attracted a surge of research attention [20], [34], [51], [55]. By proving that DeepWalk, a state-of-the-art network representation method, is equivalent to matrix factorization, text-associated DeepWalk (TADW) is proposed to incorporate the text features of vertices in the embedding space under the matrix factorization framework [51]. TADW showed better performance by incorporating the node textual information. Nevertheless, the first-order proximity is neglected in the matrix factorization process. Based on the objective function of TADW, HSCA introduced a proposed regularization term that penalizes the distance between connected vertices in the embedding space [55]. Given the basic idea of DeepWalk, TriDNR is proposed to separately learn embeddings from node–node structure relationship, node–word correlation, and label–word correspondence. Nevertheless, all these works were relied on the shallow models of matrix factorization-based variants and DeepWalk variants [34]. Attributed network embedding has also been considered for the incomplete graph with the multiview learning framework [25], [52]. A recently proposed model SNE [28] aggregated more informative representations from the ID embedding and the attribute embedding, both of which are learned through the multilayer neural networks. Hence, we argue that the patterns in attributed networks are complex and hidden, and we aim to exploit the hidden patterns in attributed networks with deep learning-based models.

C. Deep Neural Networks

Deep neural networks develop algorithms to automatically learn the nonlinear, deep complex features of objects, and have achieved success in fields like image classification [37], natural language translation [7], and so on. Among all deep learning models, autoencoder and its variants provide unsupervised approaches for feature engineering, and have also been adopted in many network-based applications. Researchers proposed an autoencoder-based approach for learning node representations in sparse networks [54]. The proposed method showed comparable results compared to state-of-the-art models. Nevertheless, it only considered the first-order proximity of nodes in the network without any global information. DNGR [6] applied the stacked denoising autoencoder to generate low-dimensional vertex representations. SDNE is a semi-supervised deep autoencoder model by enforcing the first-order proximity as the supervised information in the embedding space for network embedding without any attribute information [48]. DRNE [45] is designed with a layer normalized LSTM model to learn node representations with regular equivalence information preserves. As the graph structure evolves over time, DepthLGP, a deep generative model is proposed for dynamic network embedding [29]. Borrowing the success of these previous works, we provide one of the first few attempts to use deep learning models for attributed network embedding. In this paper, we focus on the most common case when the static network structure and network attributes are available, and leave the problem of how to model dynamic network embeddings as a future work.

TABLE I
SUMMARY OF NOTATIONS

Notations	Description
G	The attributed network
V, E	Set of vertices, edges
$\|V\|$, $\|E\|$, n	Number of vertices, edges, attributes
S	Structure adjacency matrix
A	Structure transition matrix
X$\subseteq \mathbb{R}^{\|V\|\times n}$	Attribute information matrix
R	Attribute proximity matrix
P^t	The t^{th} degree proximity matrix
Q	Enhanced attributed network matrix
d	Dimension of the learned representations
$H \in \mathbb{R}^{\|V\|\times d}$	Embedding matrix

III. PROBLEM DEFINITION

In this section, we first introduce the definition of attributed network embedding and define some important terminologies. The main notations used in this paper are listed in Table I.

Definition 1 (Attributed Network): An attributed network is denoted as $G = (V, E, X)$, where $V = \{v_1, v_2, v_3, \ldots, v_{|V|}\}$ denotes the set of nodes and $E \subseteq \mathbb{R}^{V\times V}$ denotes a set of edges. $e_{i,j}$ represents the connection between v_i and v_j. Each node $v_i \in V$ is associated with an n-dimensional attribute vector x_i. Matrix $X = [x_1 : x_2 : x_3, \ldots, x_n] \in \mathbb{R}^{n\times|V|}$ captures all binary features of the nodes.

Given an attributed network G, we can obtain a structure adjacency matrix S, where the (i, j)th entry of S represents the link information between v_i and v_j. For an unweighted graph, we have $S_{i,j} = 1$ if there exists an edge from v_i to v_j, and define it as 0 if there is no edge. For a weighted graph, if there exists an edge from v_i to v_j, $S_{i,j}$ denotes the detailed weight between them.

Definition 2 (Attributed Network Embedding): Given an attributed network $G = (V, E, X)$, the task of attributed network embedding is to represent each vertex $v_i \in V$ into a low-dimensional space with informative and continuous representation, where structure proximity in E and attribute proximity in X are preserved.

For better understanding, we illustrate an instance of attributed network embedding in Fig. 1, where the left part shows an attributed network, and the right part depicts the embedding of the nodes. For example, as (v_3 and v_5) have strong bonds in the structure space, the embeddings of them are encouraged to be similar in the embedding space, so as

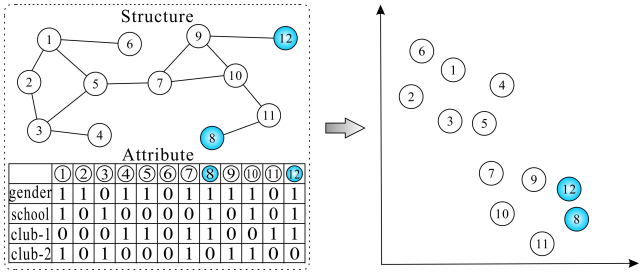

Fig. 1. Illustration of attributed network embedding. The input is an attributed network and the output is the low-dimensional embeddings of the nodes. The colored nodes share common attributes and have similar representations in the embedding space.

the vertex pairs of (v_1 and v_6) and (v_9 and v_{12}). This structure preserving property is commonly adopted by most network embedding models. In the attributed network embedding, vertex pair (v_8 and v_{12}) are close to each other as they have similar attributes, though they are not directed connected.

Definition 3 (Structure Transition Probability): The structure transition matrix $A \in \mathbb{R}^{|V|\times|V|}$ denotes the probability of direct transitions between each pair of vertices. At each time, vertex v_i will connect with v_j with a certain probability, which is defined as $A_{i,j}$ and determined by the affinity score of vertex pair (v_i, v_j). Otherwise, if there is no connection between v_i and v_j, the structure transition probability $A_{i,j}$ equals 0.

Structure transition matrix A shows the transition probability between nodes within one step, which can also be regarded as a rescaled adjacency matrix S whose rows are normalized. So, we can learn A as

$$A_{i,j} = \frac{S_{i,j}}{\sum_{v_k \in V} S_{i,k}}. \tag{1}$$

In the above equation, $A_{i,j}$ denotes the transition probability from v_i to v_j. Thus, the structure transition matrix A should satisfy the following constraints:

$$\begin{aligned} &1:\ 0 \le A_{i,j} \le 1 \\ &2:\ \forall i \in [1, 2, 3, \ldots, |V|], \quad \sum_{j=1}^{|V|} A_{i,j} = 1. \end{aligned} \tag{2}$$

Definition 4 (Attribute Proximity Probability): The attribute proximity probability represents the association between vertices evidenced by their corresponding attributes, which is related to the attribute proximity. All the attribute proximity probabilities can be represented in an attribute proximity matrix R, where $R_{i,j}$ is determined by the similarity between x_i and x_j, i.e., the attribute vector representations of v_i and v_j. The more common attributes two vertices share, the larger attribute proximity probability between them.

To better obtain attribute proximity probability from heterogeneous and highly diverse attributes matrix X, we first set matrix B to denote the proximity nodes in G evidenced by attributes. Without the loss of generality, $B_{i,j}$ is computed as

$$B_{i,j} = \frac{|\gamma \odot x_i \odot x_j|}{|\gamma \odot (x_i + x_j - x_i \odot x_j)|} \tag{3}$$

where x_i is the one-hot encoding attribute vector representation of node v_i and x_j denotes the one-hot encoding attribute representation of v_j. In fact, any attribute can be easily transformed to one-hot encoding, and this one-hot encoding is also widely applied in many real-world applications. Vector $\gamma \in \mathbb{R}^n$ controls the weights of each attribute, with $\gamma_k \in [0, 1]$ denotes the importance of the kth attribute. Considering some attributes are important, γ plays an important role in balancing different attributes.

Given the attribute proximity matrix R, the attribute proximity probability $R_{i,j}$ represents the scaled similarity of attributes between two adjacent v_i and v_j. Similar to the normalization of the structure adjacency matrix mentioned above, we perform the same procedure on node attribute proximity to obtain the normalized attribute proximity matrix, which is defined as follows:

$$R_{i,j} = \frac{B_{i,j}}{\sum_{v_k \in V} B_{i,k}} \tag{4}$$

where $B_{i,j}$ is the attribute proximity of node v_i and v_j.

IV. Proposed DANE Framework

In this section, we describe our proposed DANE framework for attributed network embedding. Given an attributed network G, its structure transition matrix A and attribute proximity matrix R as defined above, our goal is to learn a deep embedding h_i for each node v_i in a low latent space, such that the structure and attribute information are preserved. The basic idea of DANE is that, instead of operating on the structure adjacency matrix S, we borrow the random walk idea of this attributed network to build an enhanced matrix $Q \subseteq \mathbb{R}^{|V|\times|V|}$ with structure transition matrix A and attribute proximity matrix R. With this enhanced matrix Q, we could use the unsupervised deep learning model to learn the embeddings of each node to capture the complex patterns of this attributed network G. Given this basic idea, we present the architecture of DANE in Fig. 2, which consists of three main parts.

Part 1 (Step-Based Proximity Calculation): Based on the random walk in graphs, at each step t, we build a step-based proximity transition matrix P^t that captures the proximity between nodes from t-degree of proximity with both structure transition matrix A and attribute proximity matrix R.

Part 2 (Proximity Fusion): Given the t-degree proximity matrix sequences, we introduce a fusion method that produces a proximity fusion matrix $Q \subseteq \mathbb{R}^{|V|\times|V|}$ based on various P^t calculated above. Thus, Q preserves the various orders of structure and attribute information of the attributed network.

Part 3 (Deep Embedding): As the hidden information in attributed networks is usually complex and nonlinear, given the proximity fusion matrix Q, we use a deep embedding model to get the embeddings of nodes in the final step.

Next, we introduce these three steps in detail.

A. Step-Based Proximity Calculation

The step-based proximity calculates the similarity between nodes from different steps t ($t = 1, 2, 3, \ldots$), which is similar

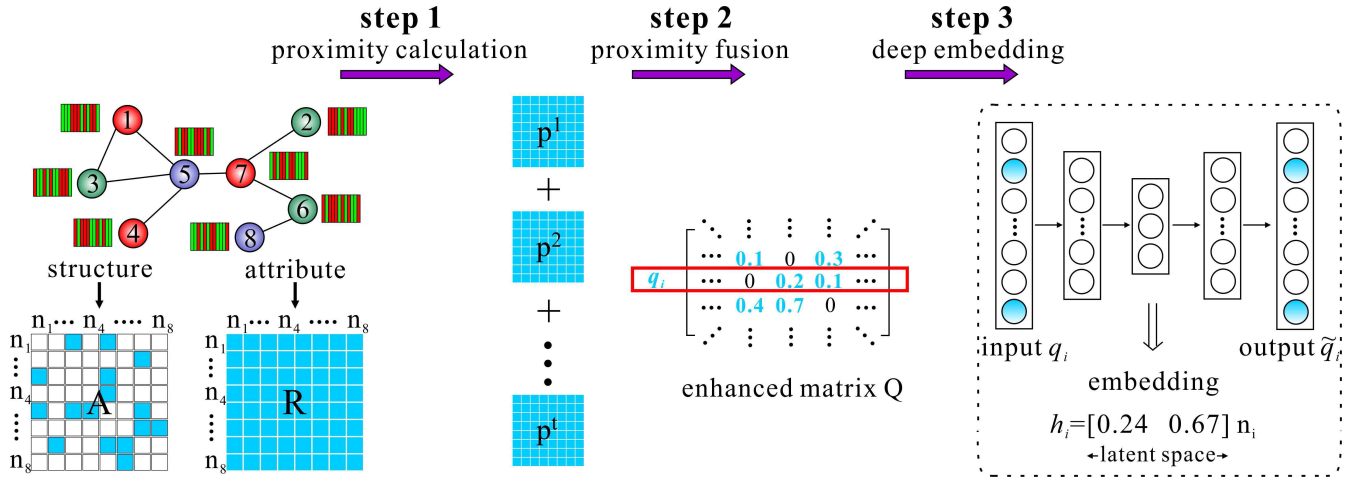

Fig. 2. Illustration of the overall framework of DANE.

to the Personalized PageRank model that is traditionally used for ranking nodes in graphs [16], [33]. Nodes connect with each other in an attributed network G is similar to that browsing the links in the Internet. By borrowing the basic ideas in Personalized PageRank, we also rely on the two assumptions for network embedding.

1) *Structure Random Walk Assumption:* It implies that node v_i will walk to node v_j with high probability if there is a connection between v_i and v_j. Two vertexes in real-world networks are always similar if they are linked by an observed edge.
2) *Attribute Proximity Assumption:* It means that node v_i will possibly be similar to node v_j if they have many similar attributes. A pair of nodes (v_i, v_j) should have more intimate relationship if they share many common attributes.

These two assumptions have been proved reasonable in many fields. For example, for the *structure random walk assumption*, if a paper cites another paper, they probably contain some common topics and should be close to each other in the embedding space. Besides, social scientists have long converged that individuals would develop friendships with others of approximately the same age and same race [23], revealing the *attribute proximity assumption*.

Given a source node v_i, we assume that it walks the next node from structure transition matrix A with probability α, and with probability $1-\alpha$ it jumps to other nodes from attribute proximity matrix R. Then, the first degree relationship between node v_i and v_j is modeled as

$$P^1_{(i,j)} = \begin{cases} \alpha A(i,j) + (1-\alpha)R_{i,j}, & \text{if} \quad e_{u,v} \in E \\ (1-\alpha)R_{i,j} & \text{otherwise.} \end{cases} \tag{5}$$

Then, for any $t \geq 1$, we can model the $(t+1)$ th step-based proximity based on the t-th proximity matrix P^t as

$$P^{t+1} = \alpha P^t A + (1-\alpha)R. \tag{6}$$

In the above equation, for the structure proximity at the $t+1$th step is the t-th step proximity matrix P^t multiplied by the graph structure matrix A. The node attribute proximity at the $t+1$th step is R as node attributes are static and do not evolve with the graph structure. Then, the final $(t+1)$th step proximity is fused by the t-th structure proximity and attribute proximity.

Please note that, in step-based proximity calculation process, the first degree proximity matrix P^1 could be treated as the linear combination of structure transition matrix A and attribute proximity matrix R, where only the first-order proximity between nodes from the structure is considered. Then, P^2 considers the second-order structure proximity between nodes, as well as the attribute proximity. The larger the t, the higher-order structure proximity, and the node attribute proximity is modeled. In such a way, we could model the local and global structure proximity, as well as the node attributes with the proximity matrix sequences: $P^1, P^2, \ldots, P^t$.

B. Proximity Fusion

In this section, we propose to merge the proximity matrix sequences: $P^1, P^2, \ldots, P^t$ into a matrix Q, that well preserves the local and global structure information, and the node attribute information. A simple idea is to simply average all these proximity matrices for merging. However, in attributed networks, the closer connections between nodes (i.e., the smaller degree), the more intimate relationship between two nodes. Intuitively, network embedding expects nodes with low-order proximity to be more influential than those of high-order proximity. So, it is better to design a weight function with the weight monotonically decreasing as step t increases. Based on this intuition, it is desirable if the proximity fusion matrix Q is defined as

$$Q = \sum_{t=1}^{T} w(t) \cdot P^t \tag{7}$$

where $w(t)$ is a decreasing function, i.e., $w(t+1) < w(t)$.

In practice, in order to preserve the information ranging from 1st degree to t-th degree appropriately, we use an exponential function adjusted with a parameter β as weighting strategies in our implementation. Please note that, any

proximity function that satisfies (7) could be used as the proximity fusion function. Then, the fusion function is indicated as follows:

$$w(t) = \beta^{-t} \tag{8}$$

where $\beta \in (0, 1)$, and the weight $w(t)$ diminishes as step t increases.

The proximity fusion method that combines different orders of proximity is widely adopted in some classical graph similarity-based metrics (e.g., the Katz index), and the recent HOPE [32] model for network embedding. Our proposed proximity fusion method differs from these works as we focus on the attributed network that considers both the structure proximity and attribute proximity, while these previous works mainly focused on the graph structure without attribute proximity modeling.

C. Deep Network Embedding

Since the fusion matrix Q contains the local and global structure, as well as the node attribute of the attributed network G, in this step, we focus on generating low-dimensional deep embedding vectors from Q with deep models. Specifically, stacked denoising autoencoder, a popular unsupervised deep representation learning model, is especially suitable as the deep learning model. It learns the deep representations of inputs by learning a nonlinear from inputs, then reconstructs the inputs from the deep representations. In the deep network embedding procedure of DANE, given the fusion matrix Q, our goal is to learn a mapping function $f : Q \to H \in \mathbb{R}^{|V| \times d}$, where H is the learned embedding matrix with the deep autoencoder network. An autoencoder neural network is an unsupervised model which is composed of two parts, i.e., the encoder and decoder, and the structure of autoencoder is displayed in the right part of Fig. 2. Here, we briefly introduce the two parts.

1) Encoder: The encoder part transforms input vector q_i to a hidden layer representation h_i, where q_i is the ith row of the fusion matrix Q. The encoder part can be mathematically formulated as

$$H = f(WQ + b) \tag{9}$$

where $\{W, b\}$ are the parameters of the encoder with $W \in \mathbb{R}^{d \times |V|}$, $b \in \mathbb{R}^d$. $f(\cdot)$ is the activation functions.

2) Decoder: The decoder part reconstructs the input matrix Q from the embedding matrix H, which can be formulated as

$$\tilde{Q} = g(W'H + b') \tag{10}$$

where $\{W', b'\}$ are the parameters of the decoder with $W' \in \mathbb{R}^{|V| \times d}$, $b' \in \mathbb{R}^{|V|}$. $g(\cdot)$ is the activation functions.

All the parameters are learned by solving the following optimization problem:

$$\min \sum_{i}^{|V|} \mathcal{L}(q_i, \tilde{q}_i; W, b, W', b'). \tag{11}$$

In practice, redundant information and noise are also contained in the fusion matrix Q. Denoising is a widely used strategy that is introduced to reduce noise and enhance robustness of autoencoder [46]. Stacked denoising autoencoder works by randomly dropping out the input before encoding in the training step. In DANE, we perform deep embedding using stacked denoising autoencoder with sparse inputs, then our model could better reconstruct the input under a noisy or partial input. Let $m^{(k)} \in \{0, 1\}^{|V|}$ denote the random binary mask that dropouts the input vector. The process of input corrupted by noise is then defined as $q_i^k = q_i \odot m^{(k)}$. As a result, the revised objective function with dropout is shown as follows:

$$\min_{\Theta} \mathcal{L}(\mathbf{Q}, \widetilde{\mathbf{Q}}) = \sum_{i=1}^{|V|} \|\left(\widetilde{q}_i^k - q_i^k\right)\|_2^2 + \frac{\lambda}{2} \cdot \left(\|W\|_2^2 + \|W'\|_2^2\right) \tag{12}$$

where $\Theta = [W, b, W', b']$ is the parameter set of the deep autoencoder.

We regularize the learned parameters to control the model complexity so as to prevent overfitting. Squared $\mathcal{L}_2$-norm is used in DANE. After building the basic architecture, we will fine-tune the parameters using the stochastic gradient descent.

D. Model Learning and Discussion

In fact, the above proposed objective function is differentiable. Therefore, similar as many deep learning-based models, we could implement DANE with TensorFlow[1] to jointly train model parameters by performing the stochastic gradient descent [1].

1) Complexity Analysis: Our proposed DANE framework is composed of three parts: 1) proximity calculation; 2) proximity fusion; and 3) deep embedding. Given a truncated step T, the time complexity of the first part is $O(T|V|^3)$, where $|V|$ is the number of vertices. The time complexity proximity fusion step is $O(|V|^2)$. For deep embedding part, the time complexity is $O(d|E|)$, where d is the maximum dimension of the hidden layer, and E is the number of edges. Since the edges are very sparse ($O(|E|) \ll O(|V|^3)$), the total time complexity of DANE is $O(T|V|^3)$.

2) Dealing With Incomplete Graph: The proposed DANE framework is set with the assumption that both the network structure and node attributes are available. In the real world, the attributed graph may be incomplete, with missing node links or missing node attributes. For example, in the social network, some users are reluctant to complete their profile information, leading to the incomplete node attributes of these users. DANE is also flexible to this situation. By adapting the proximity calculation step as shown in (5), DANE fuses the structure and attribute proximity of each node with a balance parameter α. For each node, when the node attribute is not available, $\alpha = 1$ for this node, showing that the embedding of this node preserves the structure information. In contrast, if a node only has the node attributes without any structure information, $\alpha = 0$ for this node, denoting the node embedding purely relies on the attribute information.

[1] https://www.tensorflow.org/

TABLE II
STATISTICS OF THE DATASETS

Dataset	Nodes	Edges	Attributes	Labels
# BlogCatalog	5196	171,743	8189	6
# Flickr	7575	239,738	12047	9
# Flickr3	2489	40,157	4843	3
# Email-Eu-core	1005	25,571	42	-

V. EXPERIMENTS

A. Experimental Setup

Three public networks, i.e., BlogCatalog,[2] Flickr, and Email[3] are adopted for evaluation. All of them have been used in the previous work [24], [26] and are publicly available. Table II shows the statistics of these datasets and the detailed descriptions of them are listed as follows.

1) *BlogCatalog [26]* is a social network, where people can post blogs. Bloggers can follow others to form a network. The keywords in users' blog descriptions are considered as his/her attribute information. In this online social network, every user is labeled by categories, which can be used as the label of the blogger. Then, it can be evaluated on node classification application.
2) *Flickr [26]* is an image and video hosting website, where users interact with each other via photograph sharing. The network is formed by the following relationships among users. The list of tags specified by every user is considered as his/her attribute information. There are nine groups on this website, and each user could join one group. Each user's group information could be regarded as the label information of this user.
3) *Flickr3* is a subset of Flickr dataset. We choose users from three groups in Flickr as a new dataset. The three different groups that users joined are presented as labels, which can be used in visualization task.
4) *Email-Eu-Core [24]* is generated by email data from a large European research institution. Members connect with others through emails and the department they work for are regarded as the attributes.

We compare our proposed DANE framework with the following five network embedding methods. The properties of the baselines are summarized in Table III and the detailed descriptions are listed as follows.

1) *TADW [51]:* It is an algorithm that utilizes both network and context attribute information under the framework of matrix factorization to learn network representation.
2) *DNGR [6]:* It uses a random surfing model to capture global structure information and calculates the PPMI matrix from probabilistic co-occurrence matrix. Then, a stacked denoising autoencoder is used to learn low-dimensional vertex representations from the PPMI matrix.
3) *LINE [43]:* It learns two embedding vectors for each node from the first-order and second-order proximity of the network, respectively. The embedding vectors are concatenated as the final representation for each node.
4) *DeepWalk [35]:* It learns representations by employing truncated random walks on the plain graph and using skip-gram with hierarchical softmax to analyze the walking tracks.
5) *AE+A:* It concatenates linkage and attribute information into one matrix, and conducts an autoencoder model [38] on the joint matrix for mapping them into low-dimensional vectors.

[2]http://people.tamu.edu/ xhuang/Code.html
[3]https://snap.stanford.edu/data/email-Eu-core.html

B. Parameter Settings

In our proposed DANE framework, we use stacked denoising autoencoders to generate compressed vectors from the constructed matrix Q. The structure of the deep embedding varies with different datasets. We detail dimensions of each layer in the stacked autoencoder of different datasets, which are listed in Table IV.

Regarding the choice of the activation function in the hidden layers of autoencoder, we have tried rectified linear unit, hyperbolic tangent function (tanh), and sigmoid function. As these three functions show similar performance, we choose the sigmoid function as a representative. Hence, all the neurons are activated by the sigmoid function. The parameters α and β are tuned by using grid search on the validation set. We use a grid search among the following parameters: $\alpha \in \{0.95, 0.9, 0.85, 0.8, 0.75, 0.7\}$ and $\beta \in \{0.98, 0.96, 0.94, 0.92, 0.9\}$ to find the optimal parameters, where α denotes the random walk preference ratio between structure information and attribute information and β controls the downtrend of higher-order information.

C. Experimental Results

In this section, we report the results of DANE against each baseline. Three classic applications are used to evaluate the effectiveness of node embedding results: 1) multilabel classification; 2) link prediction; and 3) visualization.

1) Multilabel Classification: Multilabel classification is an important task that aims to predict the labels of unlabeled vertices. The representations for the vertexes are generated from the network embedding methods and are used as features to classify each vertex into a set of labels. For the multilabel classification task, similar to many other works, we adopt Micro-F1 score and Macro-F1 score for evaluation [43], [56].

First, we define TP_i, FP_i, and TN_i as the *true positive*, *false positive*, and *true negative* of label i. Suppose M is the total number of categories, recall ρ and precision π are obtained by summing over all individual decisions

$$\pi = \frac{\sum_{i \in M} \mathrm{TP}_i}{\sum_{i=1}^{M} (\mathrm{TP}_i + \mathrm{FP}_i)}, \quad \rho = \frac{\sum_{i \in M} \mathrm{TP}_i}{\sum_{i=1}^{M} (\mathrm{TP}_i + FN_i)}.$$

Micro-F1 tends to be dominated by classifier's performance on common categories is then computed as

$$\text{Micro-F1} = \frac{2\pi\rho}{\pi + \rho}.$$

Macro-F1 is obtained by taking the average of F-measure values for each category as

$$\text{Macro-F1} = \frac{\sum_{i \in M} F_i}{M}$$

where F_i is the F1-measure for the label i.

TABLE III
SUMMARY OF THE BASELINES

Algorithms	First-order Proximity	Second-order Proximity	Higher-order Proximity	Attributes	non-linearly
# DANE		√	√	√	√
# TADW [51]		√	√	√	
# DNGR [6]		√	√		√
# LINE [43]	√	√			
# DeepWalk [35]		√	√		
# AE+A [38]	√			√	√

TABLE IV
STRUCTURE OF THE STACKED AUTOENCODER OF DIFFERENT DATASETS

Dataset	nodes in each layer
BlogCatalog	5196-1200-300
Flickr	7575-1500-300
Flickr3	2489-600-150
Email-Eu-core	1005-240-60

We first learn embedding vectors from training datasets (all links and attributes) under different models. After that, we use the LibLinear package [10] to train the classifiers, which is widely adopted in training one-versus-rest logistic regression classifiers. We run this process for 50 rounds and report the averaged Micro-F1 and Macro-F1 measures. For each round, we randomly sample the vertices ranging from 10% to 90% as labeled nodes and use these samples for training. The remaining vertices are used for evaluation. Since the BlogCatalog and Flickr datasets have the labels of each vertex, we show the performance of these two datasets in Table V. From the results, we have the following observations.

1) First and foremost, as shown in Table V, DANE achieves the best performance among all the baselines under all settings. It empirically demonstrates that DANE could learn better network embeddings compared to the baselines.
2) Focusing on methods that account for attributes (i.e., DANEm and TADW), we find these two methods perform better than DNGR and DeepWalk, as the latter two methods only exploit network structure without any attribute modeling. In both datasets of BlogCatalog and Flickr, DNAE and TADW achieve more than 10% gain over DeepWalk and DNGR in Macro-F1 and Micro-F1 scores, demonstrating the significant improvement of incorporating attribute information into network representation learning.
3) By taking advantage of high-order proximity in the network structure, DANE achieves more than 23% improvements than AE+A. Although attributes information is taken into consideration, AE+A is the worst among the network embedding methods, especially when the training percentage decreases from 30% to 10%. It demonstrates that the higher-order proximity is of great importance in preserving network information for label classification especially when the dataset is very sparse.
4) Our proposed framework always performs the best, followed by the baseline of TADW. In fact, TADW exploited the high-order proximity and attribute proximity. We guess a possible reason is that our proposed DANE framework modeled the complex patterns in the attributed network with deep neural networks, while the TADW relied on the shallow model of matrix factorization.

2) Link Prediction: In this section, we concentrate on the link prediction task which assesses the ability of learned representations in reconstructing network structure [28]. We use *Precision@k*, *NDCG@k*, *Recall@k*, and *AUC*, which are widely used in link prediction [54] and recommendation [17], [39] to judge the ranking quality. Their definitions are listed as follows.

1) *Precision@k* is defined as the ratio of true predicted links selected for top-k edges. The *Precision@k* of user v_i can be calculated as follows:

$$Precision@k(v_i) = \frac{|\{v_j|e_{i,j} \neq 0, \text{index}(v_j) \leq k\}|}{k}$$

where index(v_j) is the ranked index of predicted edge $\tilde{e}_{i,j}$ to v_i and $e_{i,j} \neq 0$ indicates there is a link between v_i and v_j.
2) *NDCG@k* assigns higher importance to results at top ranks, scoring successively lower ranks with marginal fractional utility

$$NDCG@k(v_i) = Z_k \sum_{j=1}^{k} \frac{2^{r_j} - 1}{\log_2(j+1)}$$

where Z_k is the normalizer to ensure the perfect ranking has a value of 1 and r_j is the graded relevance of node at position j. In this paper, $r_j = 1$, if there is an observed link between v_i and v_j, and 0 otherwise.
3) *Recall@k* is the proportion of correctly predicted links found in the top-k link prediction list. The *Recall@k* of user v_i is defined as follows:

$$Recall@k(v_i) = \frac{|\{v_j|e_{i,j} \neq 0, \text{index}(v_j) \leq k\}|}{|\{v_j|e_{i,j} \neq 0\}|}$$

where index(v_j) is the ranked index of predicted edge $\tilde{e}_{i,j}$ to v_i.
4) *AUC* is defined as the *area under the ROC curve*. It is a summary measure with good stability that essentially average accuracy across the spectrum of test values. The trivial *AUC* of a random guess method is 0.5 and the larger the value, the better the performance.

In order to evaluate the overall performance of network representations in link prediction, we randomly hold out 10% links as the test dataset, 10% as the validation set, and the remaining 80% links are used for training. Table VI shows

TABLE V
MULTILABEL CLASSIFICATION ON BLOGCATALOG AND FLICKR WITH DIFFERENT ALGORITHMS

BlogCatalog Dataset										
Model	Macro-F1					Micro-F1				
	90%	70%	50%	30%	10%	90%	70%	50%	30%	10%
DANE	**0.8485**	**0.8385**	**0.8273**	**0.8169**	**0.7787**	**0.8503**	**0.8456**	**0.8381**	**0.8237**	**0.7861**
TADW	0.8271	0.8238	0.8121	0.7957	0.7429	0.834	0.8308	0.8197	0.8058	0.7588
DNGR	0.7031	0.7027	0.6969	0.6892	0.6709	0.7082	0.7069	0.7011	0.6879	0.6765
LINE	0.7058	0.6997	0.6939	0.6818	0.6363	0.7135	0.7072	0.7009	0.6884	0.6421
DeepWalk	0.6887	0.6812	0.6745	0.6552	0.5747	0.6981	0.6906	0.6832	0.6625	0.5796
AE+A	0.6194	0.6132	0.6032	0.5775	0.501	0.6292	0.6217	0.6113	0.5861	0.5121
Flickr Dataset										
Model	Macro-F1					Micro-F1				
	90%	70%	50%	30%	10%	90%	70%	50%	30%	10%
DANE	**0.7319**	**0.7331**	**0.7231**	**0.7056**	**0.6486**	**0.7386**	**0.7367**	**0.7266**	**0.7097**	**0.6548**
TADW	0.7171	0.7115	0.7022	0.6836	0.6343	0.7192	0.7131	0.7036	0.6847	0.6346
DNGR	0.6045	0.5893	0.5707	0.5444	0.4977	0.6105	0.5942	0.5743	0.5472	0.5018
LINE	0.5774	0.5738	0.5676	0.5552	0.5180	0.5893	0.5836	0.5772	0.5651	0.5278
DeepWalk	0.5540	0.5495	0.5381	0.5147	0.4256	0.5656	0.5601	0.5479	0.5233	0.4295
AE+A	0.5227	0.5187	0.5042	0.4791	0.4097	0.5348	0.5291	0.5147	0.4902	0.4225

TABLE VI
LINK PREDICTION PERFORMANCE ON FLICKR AND EMAIL-EU-CORE WITH DIFFERENT ALGORITHMS

Model	Flickr Dataset				Email-Eu-core Dataset			
	Precision@3	NDCG@3	Recall@3	AUC	Precision@3	NDCG@3	Recall@3	AUC
DANE	**0.4576**	**0.4912**	**0.1839**	**0.9363**	**0.3943**	**0.4183**	**0.4297**	**0.9515**
TADW	0.4328	0.4583	0.1739	0.9358	0.3793	0.4001	0.4139	0.9314
DNGR	0.4261	0.4565	0.1713	0.9233	0.3844	0.4086	0.4194	0.9135
LINE	0.3827	0.4123	0.1545	0.9098	0.3674	0.3874	0.3986	0.9087
DeepWalk	0.4197	0.4462	0.1687	0.9319	0.3742	0.3969	0.4084	0.9256
AE+A	0.3789	0.3964	0.1523	0.9029	0.3628	0.3894	0.361	0.9149

Precision@3, *NDCG@3*, *Recall@3*, and *AUC* score of DANE and baselines on Flickr and Email-Eu-core datasets. A higher value indicates a better performance. The best performance is highlighted in bold. Based on this table, we have the following observations.

1) The results show that the representations learned by DANE have higher scores compared to the baselines under all measures, which demonstrates that the learned network representations of our method have much better power in the link prediction task. For instance, on Flickr, our method achieves at least 3.29% improvement than baselines for *NDCG@3*.
2) The effectiveness of modeling attribute and high-order proximity information is highly demonstrated in Table VI. For example, DNGR generally outperforms LINE, for the reason that higher degree information can be captured in DNGR. And DANE achieves better performance than DNGR, which shows the positive effects of incorporating attributes into the network embedding process. It demonstrates that capturing sufficient information for link prediction can partially alleviate data sparsity issue and reach a better performance.

3) Visualization: Another way of assessing the quality of the network embeddings is through visualization. We conduct visualization experiments by following Tang *et al.* [43] and comparing the performance of our model with the baselines on Flick3. We choose Flick3 as it is a part of Flickr dataset with three classes, and it is suitable for visualization. We first learn low-dimensional node embeddings from the original dataset with different models. Then the learned graph representations are fed as the input to t-SNE tool [30]. As a result, nodes are mapped as two-dimensional (2-D) vectors and graphs can be visualized on the 2-D space. Nodes with the same label share the same color, and a good visualization performance is that points of the same color are close to each other. Fig. 3 shows the visualization figure with different embedding methods.

Many attributed networks have community structure, which shows dense vertexes–vertexes connections and highly similar properties within the same categories, but relation among vertices outside the community is sparse. Then, the clustering structure can be learned with different embedding approaches and create meaningful visualizations. Fig. 3 compares the visualization results with different modes. It is obvious that AE+E achieves the worse performance as the points belonging to different categories are mixed with each other. The clusters of three categories are formed in LINE, DeepWalk, and DNGR. However, some points with different labels are still mixed with each other, especially the points in red. For TADW and DNGR, the results look better because three groups are formed and the boundaries of different groups are very clear in DANE. We guess the reason is that the abundant attribute information which can reflect user' habits is taken into consideration in TADW and DANE.

D. Analysis on Model

In this section, we investigate the performance of DANE under different parameter settings. Specifically, we evaluate

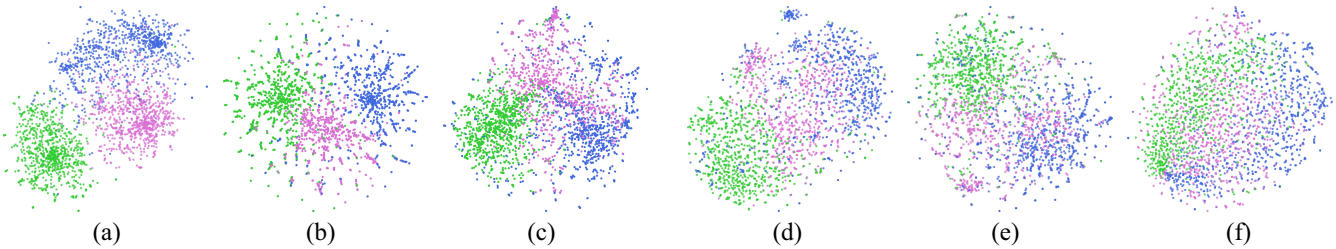

Fig. 3. Visualization of Flickr3 network by different algorithms. (a) DANE. (b) TADW. (c) DNGR. (d) LINE. (e) DeepWalk. (f) AE+A.

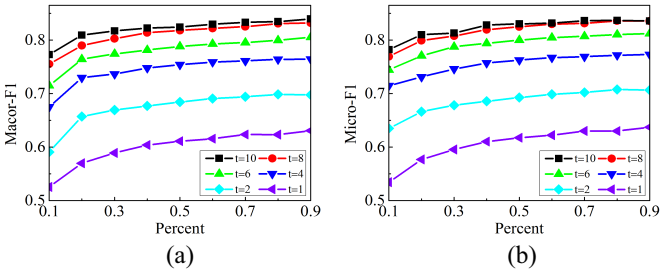

Fig. 4. Performance comparison on different t with weighting parameter $\beta = 0.96$. (a) Macro-F1. (b) Micro-F1.

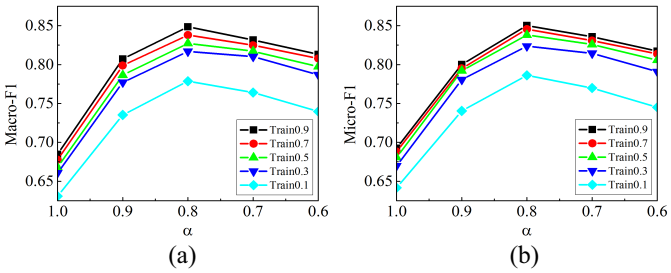

Fig. 5. Performance comparison on different α with different proportions of train-set. Train0.9 expresses 90% nodes are sampled as train set. (a) Macro-F1. (b) Micro-F1.

the effectiveness of DANE from the impact of step size T, the impact of balance point α, and the impact of embedding dimension d. We use the node classification on BlogCatalog with Macro-F1 and Micro-F1 scores as an example to show the results. The trend is similar in Flickr and Email-Eu-core datasets.

1) Impact of Step Size T: Step size T captures the global higher-order structure information. As T increases, more global structure is captured. In order to present the impact of high-order proximity intuitively, performances over a series of step sizes T are shown in Fig. 4. Different degrees of proximity are fused by weighting strategies of (7) with $\beta = 0.96$, which has been previously learned. When $T = 1$, the performance is totally determined by the observed first-order relationship between vertexes. As shown in the figure, step size $T = 2$ has an obvious improvement over step size $T = 1$, which indicates the importance of second-order proximity. The second-order proximity captures the neighborhood information between nodes, and two nodes have larger second-order proximity if they share similar neighborhood structure even though they are not directly connected. With the increase of T, higher degree information is captured, And the performances of $T = 10$ is much better than $T = 1$ and $T = 2$. However, the improvement margin over different step sizes sharply decline when the step increases from $t = 8$ to $t = 10$. We suspect that weaker information stored in t-th degree relational information can account for this, as mentioned in Section IV-B.

2) Impact of Balance Point α: Parameter α balances the weight of structure information and attribute information (5). The smaller the α, the more attention is paid to attribute proximity during the random walk process. In this section, we investigate the effects of parameters α and show the results in Fig. 5. When $\alpha = 1$, the performance is totally determined by the structure proximity. It is obvious that the performance of $\alpha = 0.8$ is better than that of $\alpha = 1$, which demonstrates that both structure information and attribute proximity are essential for network embedding. When α decreases from 1 to 0.8, the more attention is paid to attribute information and the performance of DANE improves. However, when α is lower than 0.8, the performance becomes worse as fewer attention is paid to the structure proximity. The result reveals $\alpha = 0.8$ is the equilibrium point of the structure and attribute proximity in the BlogCatalog dataset.

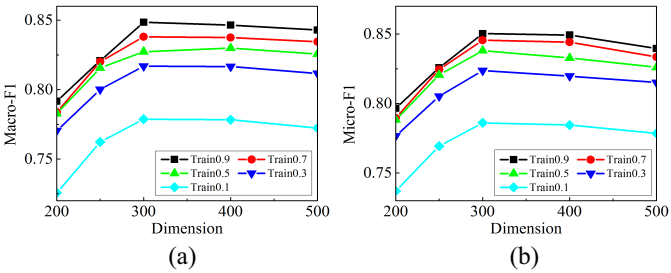

Fig. 6. Performance over different dimension d on different proportion of train-set. Train0.9 expresses 90% nodes are sampled as train set. (a) Macro-F1. (b) Micro-F1.

3) Impact of Embedding Dimension d: We finally study how the dimension of embedding vectors affects the node classification performance. We vary the dimension d from 200 to 500. The Macro-F1 and Micro-F1 scores of different ratios of train dataset are shown in Fig. 6. As we can see, initially the performance raises with the increase of d. This is intuitive as smaller values of embedding dimensions are hard to encode abundant information that is contained in the fusion matrix Q. It is obvious that when $d = 300$, the performance is good. However, when the number of dimensions continuously

increases, the performance of DANE keeps stable and drops slowly. The reason is that too large a number of dimensions may introduce noises and deteriorate the performance.

VI. Conclusion

In this paper, we proposed to tackle the problem of attributed network embedding, which involved learning low-dimensional representations of nodes that preserve both the structure and the attribute information. We proposed a DANE framework that tackled the *data sparsity*, *structure and attribute preserving*, and *nonlinearly* patterns of attributed network embedding in a unified framework. Specifically, the DANE framework is composed of three steps. First, a step-based random walk is proposed to capture the interaction between network structure and node attributes from various degrees of proximity. Then, we constructed an enhanced matrix representation of the attributed network by summarizing the various degrees of proximity. In the third step, we designed a deep neural network to exploit the complex, and nonlinear patterns in the enhanced matrix for network embedding. We conducted extensive experimental results on various datasets, and the results clearly showed the superiority of our proposed DANE framework compared to the state-of-the-art baselines.

In the future, we would like to explore and extend the proposed DANE framework for attributed network embedding models from the following two directions. First, we would improve the efficiency of DANE by learning to hash techniques, such that it could be applied to large-scale industrial scenarios. Second, as the network structure evolves over time, new edges come and old edges disappear. The incoming nodes may be incomplete with missing links or missing attributes. We plan to design the incremental algorithms for attributed network embedding as a future direction.

References

[1] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement.*, 2016, pp. 265–283.

[2] L. Backstrom and J. Leskovec, "Supervised random walks: Predicting and recommending links in social networks," in *Proc. ACM 4th Int. Conf. Web Search Web Data Min.*, 2011, pp. 635–644.

[3] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, Jun. 2003.

[4] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1616–1637, Sep. 2018.

[5] S. Cao, W. Lu, and Q. Xu, "GraRep: Learning graph representations with global structural information," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manag.*, 2015, pp. 891–900.

[6] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 1145–1152.

[7] K. Cho *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Nat. Lang. Process.*, 2014, pp. 1724–1734.

[8] L. Cui, J. Wu, D. Pi, P. Zhang, and P. Kennedy, "Dual implicit mining-based latent friend recommendation," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published.

[9] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 8, pp. 1–19, Aug. 2015.

[10] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Aug. 2008.

[11] J. R. Firth, "A synopsis of linguistic theory 1930–1955," in *Studies in Linguistic Analysis*. Oxford, U.K.: Blackwell, 1957, pp. 1–32.

[12] M. Gao, L. Chen, X. He, and A. Zhou, "BiNE: Bipartite network embedding," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2018, pp. 715–724.

[13] S. Gao, L. Denoyer, and P. Gallinari, "Temporal link prediction by integrating content and structure information," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manag.*, 2011, pp. 1169–1174.

[14] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1024–1035.

[15] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Eng. Bull.*, vol. 40, no. 3, pp. 52–74, Dec. 2017.

[16] T. H. Haveliwala, "Topic-sensitive PageRank: A context-sensitive ranking algorithm for Web search," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 4, pp. 784–796, Jul./Aug. 2003.

[17] X. He, T. Chen, M.-Y. Kan, and X. Chen, "TriRank: Review-aware explainable recommendation by modeling aspects," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manag.*, 2015, pp. 1661–1670.

[18] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proc. 40th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2017, pp. 355–364.

[19] X. He *et al.*, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.

[20] X. Huang, J. Li, and X. Hu, "Accelerated attributed network embedding," in *Proc. SIAM Int. Conf. Data Min.*, 2017, pp. 633–641.

[21] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 5th Int. Conf. Learn. Represent.*, 2017, pp. 1–14.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1106–1114.

[23] S. B. Kurth, "Friendships and friendly relations," in *Friendship as a Social Institution*. New York, NY, USA: Routledge, 2017, pp. 136–170.

[24] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Trans. Knowl. Disc. Data*, vol. 1, no. 1, p. 2, 2007.

[25] C. Li *et al.*, "PPNE: Property preserving network embedding," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2017, pp. 163–179.

[26] J. Li, X. Hu, J. Tang, and H. Liu, "Unsupervised streaming feature selection in social media," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manag.*, 2015, pp. 1041–1050.

[27] D. Lian *et al.*, "High-order proximity preserving information network hashing," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2018, pp. 1744–1753.

[28] L. Liao, X. He, H. Zhang, and T.-S. Chua, "Attributed social network embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 12, pp. 2257–2270, Dec. 2018.

[29] J. Ma, P. Cui, and W. Zhu, "DepthLGP: Learning embeddings of out-of-sample nodes in dynamic networks," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 370–377.

[30] L. V. D. Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[31] A. A. Nugraha, A. Liutkus, and E. Vincent, "Multichannel audio source separation with deep neural networks," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 24, no. 9, pp. 1652–1664, Sep. 2016.

[32] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2016, pp. 1105–1114.

[33] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the Web," Stanford InfoLab, Stanford, CA, USA, Rep. 1999-66, Nov. 1999.

[34] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-party deep network representation," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, 2016, pp. 1895–1901.

[35] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2014, pp. 701–710.

[36] D. Rafailidis and A. Nanopoulos, "Modeling users preference dynamics and side information in recommender systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 6, pp. 782–792, Jun. 2016.

[37] J. Schmidhuber, U. Meier, and D. Ciresan, "Multi-column deep neural networks for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3642–3649.
[38] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders meet collaborative filtering," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 111–112.
[39] P. Sun, L. Wu, and M. Wang, "Attentive recurrent social recommendation," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2018, pp. 185–194.
[40] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[41] C. Tan *et al.*, "User-level sentiment analysis incorporating social networks," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2011, pp. 1397–1405.
[42] J. Tang, J. Liu, M. Zhang, and Q. Mei, "Visualizing large-scale and high-dimensional data," in *Proc. 25th Int. Conf. World Wide Web*, 2016, pp. 287–297.
[43] J. Tang *et al.*, "LINE: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 1067–1077.
[44] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.
[45] K. Tu, P. Cui, X. Wang, P. S. Yu, and W. Zhu, "Deep recursive network embedding with regular equivalence," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2018, pp. 2357–2366.
[46] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, Dec. 2010.
[47] D. Wang, P. Cui, M. Ou, and W. Zhu, "Deep multimodal hashing with orthogonal regularization," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 2291–2297.
[48] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2016, pp. 1225–1234.
[49] Z. Wang *et al.*, "Discovering and profiling overlapping communities in location-based social networks," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 4, pp. 499–509, Apr. 2014.
[50] L. Wu *et al.*, "Modeling the evolution of users' preferences and social links in social networking services," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 6, pp. 1240–1253, Jun. 2017.
[51] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 2111–2117.
[52] D. Yang, S. Wang, C. Li, X. Zhang, and Z. Li, "From properties to links: Deep network embedding on incomplete graphs," in *Proc. ACM Conf. Inf. Knowl. Manag.*, 2017, pp. 367–376.
[53] T. Yang, R. Jin, Y. Chi, and S. Zhu, "Combining link and content for community detection: A discriminative approach," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2009, pp. 927–936.
[54] S. Zhai and Z. Zhang, "Dropout training of matrix factorization and autoencoder for link prediction in sparse graphs," in *Proc. SIAM Int. Conf. Data Min.*, 2015, pp. 451–459.
[55] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Homophily, structure, and content augmented network representation learning," in *Proc. 16th Int. Conf. Data Min.*, 2016, pp. 609–618.
[56] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1819–1837, Aug. 2014.
[57] S. Zhu, K. Yu, Y. Chi, and Y. Gong, "Combining content and link for classification using matrix factorization," in *Proc. 30th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2007, pp. 487–494.

Richang Hong (M'12) received the Ph.D. degree from the University of Science and Technology of China, Hefei, China, in 2008.

He is currently a Professor with the Hefei University of Technology, Hefei. His current research interests include multimedia question answering, video content analysis, and pattern recognition. He has coauthored over 60 publications in the above areas.

Mr. Hong was a recipient of the Best Paper Award in the ACM Multimedia 2010. He is a member of the Association for Computing Machinery.

Yuan He received the B.S. degree in electronic information science and technology from the Hefei University of Technology, Hefei, China, in 2016, where she is currently pursuing the M.S. degree in electronics and communications engineering.

Her current research interests include social network analysis and recommender systems.

Le Wu received the Ph.D. degree in computer science from the University of Science and Technology of China, Hefei, China, in 2016.

She is currently an Assistant Professor with the Hefei University of Technology, Hefei. She has published over 20 papers in top conferences and journals, such as AAAI Conference on Artificial Intelligence, ACM Knowledge Discovery and Data Mining, International Joint Conference on Artificial Intelligence, International Conference on Research on Development in Information Retrieval, SIAM International Conference on Data Mining, IEEE International Conference on Data Mining, ACM International Conference on Information and Knowledge Management, the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, and *ACM Transactions on Intelligent Systems and Technology*. Her current research interests include data mining, recommender system, and social network analysis.

Ms. Wu was a recipient of the Distinguished Dissertation Award from China Association for Artificial Intelligence and the Best of SDM 2015 Award.

Yong Ge received the Ph.D. degree in information technology from the Rutgers, The State University of New Jersey, New Brunswick, NJ, USA, in 2013.

He is an Assistant Professor of management information systems with the University of Arizona, Tucson, AZ, USA. He has published prolifically in refereed journals and conference proceedings, such as the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, *ACM Transactions on Information Systems*, *ACM Transactions on Knowledge Discovery From Data*, ACM Knowledge Discovery and Data Mining (ACM SIGKDD), SIAM International Conference on Data Mining, IEEE International Conference on Data Mining (IEEE ICDM), and ACM Conference on Recommender Systems. His current research interests include data mining and business analytics.

Mr. Ge was a recipient of the ICDM-2011 Best Research Paper Award. He was also a Program Committee Member at ACM SIGKDD and IEEE ICDM.

Xindong Wu (F'11) received the Ph.D. degree in artificial intelligence from the University of Edinburgh, Edinburgh, U.K.

He is currently an Alfred and Helen Lamson Endowed Professor of computer science with the School of Computing and Informatics, University of Louisiana at Lafayette, Lafayette, LA, USA. His current research interests include data mining, knowledge-based systems, and Web information exploration.

Dr. Wu is the Steering Committee Chair of the IEEE International Conference on Data Mining and the Editor-in-Chief of *Knowledge and Information Systems* (Springer) and *Advanced Information and Knowledge Processing* (Springer). He was the Editor-in-Chief of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING from 2005 to 2008. He is a fellow of American Association for the Advancement of Science.