Sparse Attributed Network Embedding via Adaptively Aggregating Neighborhood Information

Ying Chen[1], Jingwei Zheng[1], Dagang Li[2,1]✉
[1]*School of ECE, Peking University Shenzhen Graduate School, Shenzhen, China*
[2]*International Institute of Next-Generation Internet, Macau University of Science and Technology, Macau SAR, China*
Email: chen-ying@pku.edu.cn, zhengjingwei@pku.edu.cn, dagang.li@ieee.org

***Abstract*—Network representation learning (NRL), which aims to map nodes in a network into low-dimensional vectors, has attracted wide attention due to its potential on various network applications. Recently, attributed network embedding that incorporates both network structure and node attributes has shown a good performance in NRL. However, most existing methods cannot achieve promising results in sparse attributed networks because of the lack of structural information. In order to further alleviate the negative effect of sparseness, we propose a novel model named AANI (Adaptively Aggregating Neighborhood Information). Specifically, AANI exploits the information of current node and its neighborhood in a smoothing way to obtain more informative node representations. At the same time, AANI introduces attention mechanism to adaptively aggregate the smoothed information according to their respective importance. We conduct extensive experiments on three real-world datasets which demonstrate that AANI outperforms the state-of-the-art embedding methods in sparse attributed networks.**

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

I. Introduction

With the development of embedding techniques, many researchers have paid more attention to the network embedding algorithms, which aims to map nodes in a network into low-dimensional vectors. Network embedding is a feature learning method that can be applied in various network applications such as node classification [1], link prediction [2] and network visualization [3]. Previous works focus on preserving local and global structural information in the network, such as DeepWalk [7] and node2vec [8]. These methods usually perform truncated random walks on the network to generate node sequences similar with the corpus in word2vec [6]. However, other useful information in the network, such as node attributes, has been largely ignored.

Nodes associated with attributes can be easily observed in the real-world networks, which are termed as attributed networks. For example, in the Facebook network, age, gender and post contents can be treated as the attributes of the source node. Similarly, in a citation network, the appearance of important words in papers can be also treated as attributes. Node attributes can be beneficial to revealing the network structure. For example, the pattern of node attribute is often similar in the neighborhood, which is known as homophily effect. Many existing methods have demonstrated the benefits of attributes. TADW [14] first utilizes the framework of matrix factorization to incorporate the text attributes of nodes. ASNE [16] proposes a deep model preserving the structure proximity and attribute proximity simultaneously. However, most of them cannot achieve promising results when the network is relatively sparse.

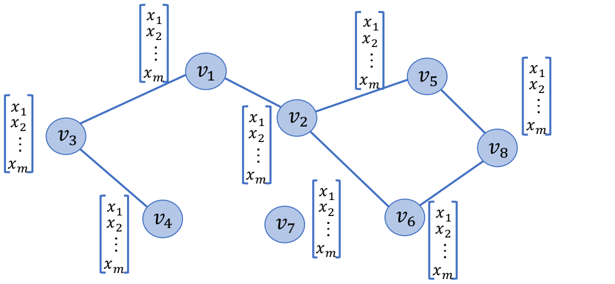


Fig. 1. A toy example on a sparse attributed network. Nodes are represented by circles and the attribute information is next to each node.

Sparseness means the lack of structural information, because there are less links between nodes. Node attributes as an important auxiliary information can alleviate the effect of sparseness on node representations. However, most existing methods can only utilize the attribute of current node but ignore the attributes of its neighborhood nodes, which cannot fully exert the potential of attribute information. For example, in Fig. 1, the methods such as AANE [15] and ASNE [16] only utilize the attribute of current node v_1 but totally ignore the attributes of its surrounding nodes. Some recent works such as SEANO [19] and ANRL [20] improve their performance by exploiting the attributes of its adjacency nodes such as v_2 and v_3. However, they cannot exploit the attributes of its higher order neighborhood nodes such as v_4 and v_5. Exploiting the higher order neighborhood attributes can further alleviate the negative effect of sparseness and obtain more informative node representations.

In this paper, we propose a sparse attributed network embedding method named AANI (**A**daptively **A**ggregate **N**eighborhood **I**nformation). We first exploit the information of current node and its neighborhood in a smoothing way. Then, we aggregate the smoothed information by attention mechanism based on their respective importance on the node representation. After obtaining the aggregated information, we feed it into a multilayer perceptron to capture more non-

✉ Corresponding Author

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

linear information. Finally, in order to capture the structural information, we perform random walks on the network and maximize the occurrence probability of current node and its context nodes. In summary, the contributions of this paper are as follows:

- We propose a sparse attributed network embedding method named AANI, which exploits the information of current node and its neighborhood to further alleviate the negative effect of sparseness.
- We introduce attention mechanism to adaptively aggregate the information of current node and its neighborhood based on their respective contribution to the node representation.
- We conduct extensive experiments on three real-world datasets with three practical tasks: node classification, link prediction and network visualization. The results demonstrate the effectiveness of the proposed model.

The rest of this paper is organized as follows. We first summarize the related work in Section II, followed by providing some problem statements in Section III. Our model and experimental results are presented in Section IV and V. A simple conclusion is presented in Section VI.

II. Related Work

A. Network Embedding

Network embedding technologies can be traced back to the graph based dimensional reduction methods, such as Locally Linear Embedding (LLE) [4] and IsoMAP [5]. These methods learn the data embedding while preserving the local manifold structure. The major issue of these methods is that they cannot be applied to large-scale networks due to the high computational complexity in calculating eigenvectors. Recently, neural networks have made great progress in many fields such as natural language processing [12] and computer vision [13]. Inspired by the success of word2vec [6], DeepWalk [7] performs truncated random walks on the network to generate node sequences and feeds them into the skip-gram model to learn node representations. Node2vec [8] extends DeepWalk by adopting a more flexible strategy to generate node sequences. Except for the methods based on random walks, LINE [9] proposes a carefully designed objective function which captures the first-order and second-order proximity to preserve both local and global network structure. GraRep [10] proposes a method that captures the *k-th* order relational information to improve the performance of node representations. Different from these works, SDNE [11] proposes a semi-supervised deep model that can exploit the highly non-linear information and capture the structure proximity at the same time.

The aforementioned methods can only utilize the structural information in the network. However, attributes of nodes are important information to reveal the network structure. It has been demonstrated that attribute information can be beneficial to the node representations. TADW [14] first proposes the text-associated DeepWalk, which can incorporate text features of nodes into the network representations. However, TADW can only deal with text attributes. AANE [15] is a distributed method that can learn a low-dimensional representation through decomposition of attributed affinity and enhancing the embedding difference between connected nodes. ASNE [16] proposes a deep model which preserves the complex relationship between network structure and node attributes. However, these methods only utilize the attribute information of current node but ignore the attributes of its neighborhood, which is insufficient in sparse attributed networks. SEANO [19] and ANRL [20] utilize the attributes of adjacency nodes to enhance the performance of node representations. However, they cannot exploit higher order neighborhood attributes which can further alleviate the negative effect of sparseness. GraphSAGE [21] generates embeddings by sampling and aggregating features from a node's neighborhood. Different from GraphSAGE, we have adopted a different aggregation strategy and introduced the attention mechanism to adaptively allocate weights based on the importance of information. There are some research works explore in a semi-supervised way to incorporate label information for attributed network embedding such as TriDNR [17] and LANE [18] . However, it's often difficult to obtain the node label information in real-world scenarios.

Furthermore, there are some efforts exploring representations in dynamic networks. DNE [22] extends skip-gram in the dynamic environment with high efficiency. JODIE [23] learns dynamic embeddings of users and items from a sequence of temporal interactions. In this paper, we focus on the static network embedding.

B. Attention Mechanism

Recently, attention mechanism has proved its effectiveness in many fields such as natural language processing [24] and computer vision [25]. Attention mechanism is inspired by the human vision mechanism that people will focus on the selective parts of targets which are more important to them. Some works have introduced attention mechanism into attributed network embedding. SANE [26] introduces an attention mechanism to adaptively weigh the strength of interactions between each node and its context nodes to solve the problem caused by the highly sparse attributes. GAT [27] proposes a neural network architecture that operates on network data through leveraging masked self-attention layers. Different from SANE and GAT, our proposed model introduces the attention mechanism to adaptively weigh the importance of information which is collected from the current node and its neighborhood.

III. Problem Statement

In this section, we will first introduce some notations in this paper, and then give the definitions of our problem. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{A})$ denotes a sparse attributed network, where $\mathcal{V} = \{v_i\}_{i=1,\dots,N}$ is the set of nodes, $e_{i,j} = (v_i, v_j) \in \mathcal{E}$ is an edge encoding the relationship between node v_i and v_j, $\boldsymbol{A} = \{\boldsymbol{a}_i\}_{i=1,\dots,N}$ encodes the attribute information and $\boldsymbol{a}_i \in \mathbb{R}^M$ is the attribute of node v_i. Sparseness indicates that $\mathcal{O}(\mathcal{E}) = \mathcal{O}(\mathcal{V})$ in the network.

Definition 1. *(Structure Proximity) Structure proximity denotes the proximity that can be observed by links. If there exists an edge between node v_i and v_j, it indicates the first-order proximity. If node v_j is within the context of v_i, it indicates the high-order proximity. The first-order and high-order proximity are both structure proximity.*

The first-order proximity indicates the direct proximity in the network which can be viewed as the local proximity. The high-order proximity indicates the neighborhood similarity which can be viewed as the global proximity. The nodes sharing similar neighbors are also similar, although they are not directly connected. A popular way to capture the high-order proximity is performing random walks on the network to generate node sequences. Like [7], [8], if two nodes appear in a same node sequence, they are viewed as appearing the same context. In the remaining of this paper, we use the term "context nodes" to denote both the directly connected nodes and the nodes in the same context for simplicity.

Definition 2. *(Attribute Proximity) Attribute proximity denotes the similarity of attributes between node v_i and v_j. The correlation between $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$ represents the attribute proximity.*

Definition 3. *(Sparse Attributed Network Embedding) Given a sparse attributed network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{A})$, where $\mathcal{O}(\mathcal{E}) = \mathcal{O}(\mathcal{V})$ in the network, we aim to learn each node $v_i \in \mathcal{V}$ as a low-dimensional vector $\boldsymbol{y}_i \in \mathbb{R}^d$, where $d \ll |\mathcal{V}|$. The mapping function $f : v_i \rightarrow \boldsymbol{y}_i$ can preserve the structure proximity and attribute proximity simultaneously.*

IV. Proposed model

In this section, we first introduce how to deal with the information of neighborhood in a smoothing way. Then, we will introduce our proposed model.

A. Capturing the Information of Neighborhood

In a sparse attributed network, it is hard to capture the structure proximity because of the lack of structural information in the network. Node attributes can alleviate the problem caused by sparseness. However, most existing methods only utilize the attribute of current node but ignore the attributes of its neighborhood, which is insufficient when the network is relatively sparse. AANI improves the quality of node representations through exploiting the attribute information of current node as well as its neighborhood. Furthermore, the structural information can be implicitly captured in the process of searching neighborhood.

Neighborhood attributes smoothing. In order to exploit the attribute information, we first search the neighborhood which is within *k* hops from the central node, denoted as $\mathcal{N}(v_i, k)$. Then, we average the attributes of the nodes in $\mathcal{N}(v_i, k)$ to present the *k-th* order smoothed attribute, denoted as $\boldsymbol{a}_{i,k}$. For example, in Fig. 1, the neighborhood which is one hop from node v_1 is (v_2, v_3). Then, we average the attributes of (v_2, v_3) to obtain the smoothed attribute $\boldsymbol{a}_{i,1}$. To further exploit the attribute information, we search a larger neighborhood $(v_2, v_3, v_4, v_5, v_6)$ which is within two hops from node v_1 and average their attributes to obtain $\boldsymbol{a}_{i,2}$. Similar to [19], collecting and smoothing the attributes of nodes in $\mathcal{N}(v_i, k)$ can bring two benefits. Firstly, since more attribute information can be utilized, AANI obtains more informative node representations in sparse attributed networks. Secondly, since the pattern of attribute is similar in the neighborhood, the possible attribute noise arising from the central node can be smoothed out by averaging the attributes of nodes in $\mathcal{N}(v_i, k)$. Through expanding the search range, we can exploit more attribute information in a smoother way. If a node is isolated like node v_7 in Fig. 1, we make $\boldsymbol{a}_{i,k}$ equal to $\boldsymbol{a}_i$ for any value of *k*. For the convenience of description, we use $\boldsymbol{a}_{i,0}$ instead of $\boldsymbol{a}_i$ to present the attribute of current node which can be treated as the zeroth smoothed attribute. Thus, we can obtain a more general expression of $\boldsymbol{a}_{i,k}$ as:

$$\boldsymbol{a}_{i,k} = \begin{cases} \boldsymbol{a}_i, & k = 0 \text{ or } \mathcal{N}(v_i, k) = \emptyset \\ \frac{1}{|\mathcal{N}(v_i,k)|} \sum_{v_j \in N(v_i,k)} \boldsymbol{a}_j, & \mathcal{N}(v_i, k) \neq \emptyset \end{cases} \tag{1}$$

where $v_i \in \mathcal{V}, k \in \{0, 1, \ldots, K\}$ and K is the largest search hops. Then, we can get an attribute set $s_i = (\boldsymbol{a}_{i,0}, \boldsymbol{a}_{i,1}, \ldots, \boldsymbol{a}_{i,K})$ which is prepared for the next training step. We present the details of neighborhood attributes smoothing in **Algorithm 1**.

Algorithm 1 Neighborhood Attributes Smoothing

```
Input:
    Sparse attributed network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{A})$, the largest search hops $K$;
Output:
    Attribute set $s_i$ for all nodes $v_i \in \mathcal{V}$;
for $v_i \in \mathcal{V}$ do
    Initialize an empty set $s_i$;
    for $k = 0, 1, \ldots, K$ do
        Search the neighborhood which is within k hops from node $v_i$ into $\mathcal{N}(v_i, k)$;
        Compute $\boldsymbol{a}_{i,k}$ by (1);
        Add $\boldsymbol{a}_{i,k}$ into $s_i$;
    end for
end for
return $s_i$ for all nodes $v_i \in \mathcal{V}$;
```

Neighborhood structure capturing. Searching the *k* hops neighborhood from the central node also implicitly captures the structural information. For example, when we search the neighborhood (v_2, v_3) which is one hop from node v_1, we can capture the first-order proximity according to the definition 1 in Section III, which indicates the direct links between nodes. With the growth of the value of *k*, a wider range will be searched and more structural information will be captured.

B. Framework

An overview of AANI is shown in Fig. 2. We will introduce our proposed model layer by layer.

Input layer. The input layer shown in Fig. 2 is the set $s_i = (\boldsymbol{a}_{i,0}, \boldsymbol{a}_{i,1}, \ldots, \boldsymbol{a}_{i,K})$ for all nodes $v_i \in \mathcal{V}$. The set contains the attribute information of current node and its neighborhood, and $\boldsymbol{a}_{i,k}$ presents the *k-th* order smoothed attribute.

Attension layer. The smoothed attribute information in different search ranges has different effects on the node representation. Actually, it is difficult to know which of the *k-th* order smoothed attribute is more important. For each node in the network, AANI utilizes attention mechanism to adaptively assign weights to $\boldsymbol{a}_{i,k}$. The more important to the node representation, the larger weight will be assigned. Similar to [24], for each node $v_i \in \mathcal{V}$, a positive weight $\gamma_{i,k}$ is placed on each $\boldsymbol{a}_{i,k}$ to indicate the relative importance on the node representation. Formally, the attention network can be denoted as:

$$\gamma'_{i,k} = \operatorname{ReLU}\left(\boldsymbol{v}^T \boldsymbol{a}_{i,k}\right) \tag{2}$$

where $\boldsymbol{v} \in \mathbb{R}^{M \times 1}$ is an attribute-level latent vector which is learned during the training process. We utilize the activation function $\operatorname{ReLU} = \max(0, x)$ to capture more non-linear information. Then, the attention score $\gamma'_{i,k}$, where $k \in \{0, 1, \ldots, K\}$ are normalized by the softmax function:

$$\gamma_{i,k} = \frac{\exp\left(\gamma'_{i,k}\right)}{\sum_{k=0}^{K} \exp\left(\gamma'_{i,k}\right)} \tag{3}$$

$$\boldsymbol{o_i} = \sum_{k=0}^{K} \gamma_{i,k} \boldsymbol{a}_{i,k} \tag{4}$$

The final fused attribute information $\boldsymbol{o}_i \in \mathbb{R}^M$ is then aggregated by a sum of all the *k-th* order smoothed attribute weighted by their corresponding attention weights. Specially, $\boldsymbol{o_i}$ incorporates the attribute information of current node and its neighborhood as well as their respective contribution.

Hidden Layers. Stacking multiple non-linear layers can capture more non-linear relationship and enhance the embedding quality [28]. Many existing methods such as DeepWalk and node2vec are limited to the capability of shallow networks, and failed to capture the non-linear information. Thus, we feed the fused attribute information $\boldsymbol{o}_i$ into a multilayer perceptron to capture the complex relationship between network structure and node attribtues. The representations of hidden layers are defined as $\boldsymbol{h}^{(0)}, \boldsymbol{h}^{(2)}, \ldots, \boldsymbol{h}^{(L)}$, which can be denoted as follows:

$$\begin{aligned} &\boldsymbol{h}^{(0)} = \boldsymbol{o}_i \\ &\boldsymbol{h}^{(l)} = \sigma\left(\boldsymbol{W}^{(l)} \boldsymbol{h}^{(l-1)} + \boldsymbol{b}^{(l)}\right), l = 1, 2, \ldots, L \end{aligned} \tag{5}$$

where $\sigma(\cdot)$ represents the possible activation functions such as ReLU, sigmod and tanh. $\boldsymbol{W}^{(l)}$ and $\boldsymbol{b}^{(l)}$ are the trainable parameters of the *l-th* layer, and *L* is the total number of the hidden layers. We use $\boldsymbol{h}^{(L)}$ as final node representations.

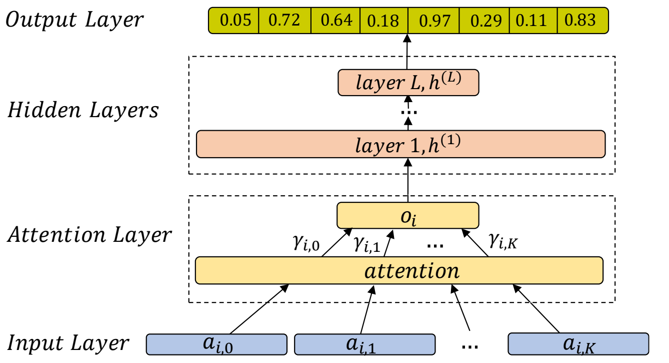


Fig. 2. An overview of our proposed model. In the input layer, $\boldsymbol{a}_{i,k}$ represents the *k-th* order smoothed attribute. Specially, $\boldsymbol{a}_{i,0}$ represents the attribute of current node.

Output Layer. Inspired by the skip-gram model used in node2vec [8] , we perform random walks on the network to generate the context corpus $C_i = \{v_{i-t}, \ldots, v_{i+t}\}$ within t window size for each node v_i. To capture the structure proximity, we utilize the final representation $\boldsymbol{h}_i^{(L)}$ of node v_i to predict the occurrence probability of its context nodes $v_j \in C_i$, which can be denoted as:

$$p\left(v_j | \boldsymbol{h}_i^{(L)}\right) = \frac{\exp\left(\boldsymbol{u}_j^T \boldsymbol{h}_i^{(L)}\right)}{\sum_{v=1}^{N} \exp\left(\boldsymbol{u}_v^T \boldsymbol{h}_i^{(L)}\right)} \tag{6}$$

where $\boldsymbol{U}$ is weight matrix for context prediction and $\boldsymbol{u}_j$ is the abstract representation when node v_j is treated as context node. Furthermore, $\boldsymbol{u}_j$ corresponds the *j-th* column in $\boldsymbol{U}$. Thus, our objective function aims to maximize the occurrence probability of node v_i and its context nodes in C_i which can be denoted as:

$$L = \prod_{v_i \in \mathcal{V}} \prod_{v_j \in C_i} p\left(v_j | \boldsymbol{h}_i^{(L)}\right) \tag{7}$$

To efficiently train the neural network, according to (6), (7) can be rewritten as:

$$L = -\sum_{v_i \in \mathcal{V}} \sum_{v_j \in C_i} \log \frac{\exp\left(\boldsymbol{u}_j^T \boldsymbol{h}_i^{(L)}\right)}{\sum_{v=1}^{N} \exp\left(\boldsymbol{u}_v^T \boldsymbol{h}_i^{(L)}\right)} \tag{8}$$

We add a regularization term in our model to prevent overfitting. Thus, the final loss function of our model can be denoted as:

$$\begin{aligned} L = &-\sum_{v_i \in \mathcal{V}} \sum_{v_j \in C_i} \log \frac{\exp\left(\boldsymbol{u}_j^T \boldsymbol{h}_i^{(L)}\right)}{\sum_{v=1}^{N} \exp\left(\boldsymbol{u}_v^T \boldsymbol{h}_i^{(L)}\right)} \\ &+ \frac{\alpha}{2}\left(\|\boldsymbol{v}\|_F^2 + \sum_{l=1}^{L} ||\boldsymbol{W}^{(l)}\|_F^2\right) \end{aligned} \tag{9}$$

where α is the ℓ_2 norm regularizer coefficient, $\boldsymbol{v}$ and $\boldsymbol{W}^{(l)}$ are weight matrices for the attention layer and hidden layers.

C. Training

Note that directly training on (9) is rather expensive, because it requires traversing all the nodes in the network when computing the occurrence probability of $p\left(v_j | \boldsymbol{h}_i^{(L)}\right)$. To reduce the computation complexity, we adopt the negative sampling strategy proposed by [29] that samples multiple negative samples according to some noisy distributions. In details, for a specific node-context pair (v_i, v_j), we then have follow objective:

$$\log \sigma \left(\boldsymbol{u}_j^T \boldsymbol{h}_i^{(L)} \right) + \sum_{n=1}^{|\text{neg}|} E_{v_n \sim P_n(v)} \left[\log \sigma \left(-\boldsymbol{u}_n^T \boldsymbol{h}_i^{(L)} \right) \right] \quad (10)$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmod function and $|\text{neg}|$ is the number of negative samples. We set $P_n(v) \propto d_v^{3/4}$ as suggested in [29], and d_v is the degree of node v.

In this way, our model preserves the structure proximity and attribute proximity in a unified framework. Furthermore, we exploit the neighborhood information in the sparse attributed network to improve the performance of node representations. To minimize the object function, we adopt stochastic gradient algorithm for optimizing (9) until the model converges. All model parameters are denoted as $\boldsymbol{\Theta}$ and the training process is summarized in **Algorithm 2**.

D. Discussions

New Nodes. Since continuous node arriving is an important situation for the evolving networks, our method provides a possible way to deal with new nodes. For the new node v_x, we can obtain the corresponding set s_x according to the Algorithm 1. Then, we can feed s_x into the finely trained model to get the representation of node v_x. Specially, since we only take the attribute information as input, our method will still work when the new node is isolated without edges.

Incomplete attribute. Node with incomplete attribute is another possible situation in attributed networks. Attributes of neighborhood as the additional information can relieve the effect of this problem. Furthermore, the attention mechanism, which adaptively aggregates attribute information from current node and its neighborhood, is also helpful for obtaining smoother representations in this case.

V. Experiments

In this section, we conduct extensive experiments with three real-world datasets to show the effectiveness of our proposed model in sparse attributed networks.

A. Datasets and Baselines

Datasets. We summarize the statistics of the three real-world datasets in Table I. The density indicates the sparseness of the network, where $d(\mathcal{G}) = \frac{2|\mathcal{E}|}{|\mathcal{V}| \times (|\mathcal{V}|-1)}$. More details are as follows:

- Cora[1] contains 2,708 papers as nodes and 5,249 citation links as edges. These papers are divided into seven categories. The attribute of each node is a binary vector of 1433 dimensions indicating the appearance of important words in the papers.
- Citeseer contains 3,312 publications of six classes such as Agents, ML, AI, DB, IR and HCI. There are 4,732 links in the network and the attributed of each node is a vector of 3,703 dimensions.

Algorithm 2 Training

```
Input:
    Sparse attributed network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{A})$, window size $b$,
    walks per vertex $\gamma$, walk length $t$, regularizer coefficient
    $\alpha$, embedding size $d$;
Output:
    Node representations $\mathbf{Y} \in \mathbb{R}^{|\mathcal{V}| \times d}$;
Construct the node context corpus $C$ by starting $\gamma$ times
    of random walks with length $t$ at each node;
Construct the set of smoothed attribute information $s_i$ for
    all nodes by Algorithm 1;
Random initialization for all parameters set $\boldsymbol{\Theta}$;
while not converged do
    Sample a mini-batch of nodes with its context and the
        set of smoothed attribute information;
    Compute the gradient of loss function based on (9);
    Update parameters in the neural network;
end while
return Node representations $\mathbf{Y} = \boldsymbol{h}^{(L)}$;
```

TABLE I
STATISTICS OF THE THREE REAL-WORLD DATASETS

Datasets	$\lvert\boldsymbol{\mathcal{V}}\rvert$	$\lvert\boldsymbol{\mathcal{E}}\rvert$	$\lvert\boldsymbol{M}\rvert$	**#density**	**#label**
Citeseer	3,312	4,732	3,703	8.63e–004	6
Cora	2,708	5,429	1,433	1.48e–003	7
Pubmed	19,717	44,338	500	2.28e–004	3

- Pubmed contains 19,717 biological papers which are divided into three classes such as "Diabetes Mellitus Experimental", "Diabetes Mellitus Type 1" and "Diabetes Mellitus Type 2". The attribute in Pubmed is represented by a 500 dimensions vector.

Baselines. In order to demonstrate the effectiveness of AANI, we compare it with several state-of-the-art methods. Since AANI incorporates both network structure and node attributes, we select the baselines from two aspects. One is the methods that only utilize network structure such as DeepWalk, LINE and SDNE. Another is the methods that utilize both network structure and node attributes such as AANE and ASNE which are competitive competitors. We also conduct two experiments named AANI/cur and AANI/noAtt to show the effectiveness of aggregating neighborhood information and attention mechanism. The details are illustrated as follows:

- *DeepWalk* [7] : DeepWalk designs a truncated random walk to generate node sequences and feeds them into the skip-gram model to learn node representations.
- *LINE* [9] : LINE preserves the first-order or second-order proximity in the network by optimizing the carefully designed objective function.
- *SDNE* [11] : SDNE proposes a deep model which can capture the highly non-linear network structure.
- *AANE* [15] : AANE is an efficient embedding method incorporating the structure proximity and attribute proximity in a distribute way.
- *ASNE* [16] : ASNE adopts a deep model to capture

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

the structure proximity and attribute proximity simultaneously.

- *AANI/cur* : AANI/cur (**cur**rent) is one of the variants of AANI. AANI/cur only utilize the attribute of current node with setting the largest search hops as 0.
- *AANI/noAtt* : AANI/noAtt (**no Att**ention) is another variant which is without the attention layer shown in Fig. 2. We take $\boldsymbol{o}_i = \frac{1}{|s_i|}\sum_{\boldsymbol{a}_{i,k}\in \boldsymbol{s}_i} \boldsymbol{a}_{i,k}$ as the input of hidden layers. AANI/noAtt demonstrates the effectiveness of attention mechanism compared with AANI.

For LINE and SDNE, we use the implementation on OpenNE[2] which is an open-source platform for network embedding. We use the implementation released by the original authors for the other methods. The parameters for baselines are tuned to be optimal. We set the embedding size *d* as 128, window size *b* as 10, walk length *t* as 80, walks per node γ as 10, negative samples as 10. For AANI, the number of neurons in each layer are shown in Table II. We set the largest search hops K as 2 in AANI and AANI/noAtt to balance the performance with computational complexity. At the end of this section, we will show the effect of the value of K on our proposed model.

B. Link Prediction

Link prediction evaluates the ability of node representations in reconstructing the network structure based on existing information. We generate the labeled dataset of edges as many methods do [8]. We randomly sample 50% existing links and an equal number of non-existing links as the positive instances and negative instances. The labeled dataset of edges is formed by both positive and negative instances. Then, we use the remaining network to train the embedding methods. After we obtain the node representations for each node, we use these representations to perform link prediction task in the labeled dataset. We rank both positive and negative instances according to the cosine similarity function. Area Under the ROC Curve (AUC) is adopted as the evaluation metric. A higher value of AUC indicates a better performance. The results of link prediction are presented in Table III. We use blue to highlight the wins and summarize the following observations:

- AANI achieves the best performance in all datasets. Compared with the best results in baselines, our method gets an improvement of 9.08% on Cora and 10.16% on Pubmed. The result shows the effectiveness of AANI in reconstructing the network structure.
- Since DeepWalk, LINE and SDNE only utilize the structural information, their performance is poor in sparse attributed networks. Interestingly, we notice that Deepwalk performs better, which mainly because that DeepWalk can explore the network structure better via truncated random walks.
- Both AANE and ASNE get an improvement compared with the methods only utilizing the structural information. These results indicate that node attributes can alleviate the netwrok sparisty. However, their performance is still poorer than AANI since they ignore the information of neighborhood nodes.

TABLE II
DETAILED NETWORK LAYER STRUCTURE INFORMATION

Datasets	Number of neurons in each layer
Citeseer	3703–3703–1000–500–128
Cora	1433–1433–500–128
Pubmed	500–500–200–128

TABLE III
LINK PREDICTION ON CITESEER, CORA AND PUBMED

Methods	Citeseer	Cora	Pubmed
DeepWalk	0.655	0.783	0.826
LINE	0.732	0.680	0.690
SDNE	0.784	0.735	0.760
AANE	0.903	0.811	0.837
ASNE	0.934	0.826	0.823
AANI	**0.943**	**0.901**	**0.922**

C. Node Classification

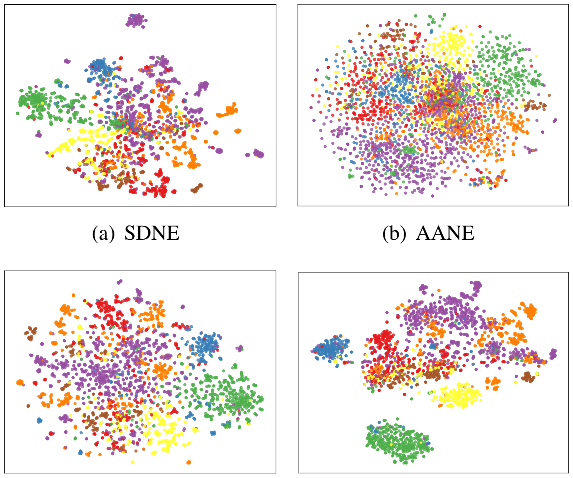
Node classification is an important task in network analysis. Similar to previous methods [7] and [8], we employ Micro-F1 and Macro-F1 as the metrics to measure the performance of node classification. The higher values of both metrices indicate better performances. After having obtained the node representations, we randomly sample 30% labeled nodes to train a SVM classifier and the rest nodes are used to test performances. We repeat this process 10 times and report the average results in Table IV. To summarize, we have the following observations:

- AANI achieves the best performance beating all the baselines. Compared with the best results, our method gets an improvement of 17.6%, 7.30% and 4.31% on Micro-F1 corresponding to Citeseer, Cora and Pubmed. The result shows that exploiting the information of neighborhood can enhance the performance of node representations.
- The methods which only utilize the structural information perform poorly such as LINE and SDNE. These methods cannot get sufficient structural information when the network is sparse. DeepWalk performs better. However, DeepWalk cannot achieve satisfactory result on Citeseer which contains many isolated nodes.
- The methods utilizing both network structure and node attributes perform better than LINE and SDNE. The results shows the benefits of attributes. However, AANE and ASNE cannot achieve promising results compared with DeepWalk on Cora and Pubmed. The result shows that only using the attribute of current node is insufficient in sparse attributed networks.
- Finally, AANI/cur performs poorly since it only utilizes the attribute of current node but ignores the attributes of its neighborhood. AANI/noAtt performs poorly since it simply averages the *k-th* order smoothed attribute in s_i

[2] https://github.com/thunlp/OpenNE

TABLE IV
NODE CLASSIFICATION ON CITESEER, CORA AND PUBMED. WE USE BLUE TO HIGHLIGHT WINS.

Datasets	**Citeseer**		**Cora**		**Pubmed**	
Evaluation	***Micro-F1***	***Macro-F1***	***Micro-F1***	***Macro-F1***	***Micro-F1***	***Macro-F1***
DeepWalk	0.586	0.540	0.795	0.783	0.812	0.798
LINE	0.458	0.426	0.686	0.673	0.766	0.749
SDNE	0.519	0.467	0.760	0.750	0.699	0.677
AANE	0.580	0.560	0.769	0.747	0.784	0.765
ASNE	0.619	0.577	0.753	0.734	0.791	0.790
AANI/cur	0.640	0.590	0.748	0.725	0.807	0.814
AANI/noAtt	0.672	0.620	0.758	0.729	0.653	0.587
AANI	**0.728**	**0.678**	**0.853**	**0.840**	**0.847**	**0.842**



(a) SDNE

(b) AANE

(c) ASNE

(d) AANI

Fig. 3. Visualization of different methods on Cora dataset.

but ignores their respective importance on the node representation. They show the necessity of all the essential components of AANI.

D. Network Visualization

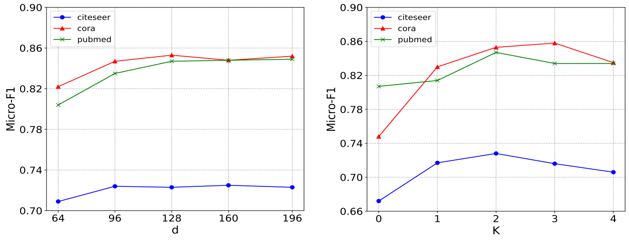
To further show the embedding result of AANI, we visualize the node representations by using t-SNE [30]. Limited by the space, we only post the results of three representative baselines on Cora. The visualization result is shown in Fig. 3.

The result of AANE is poor, which may involve the decomposition operation of attribute affinity matrix. The visualization results of SDNE and ASNE are not well separated, in which the nodes with different labels are mixed together. Compared with these methods, AANI can achieve more compact and separated clusters. Thus, AANI can achieve better performance on three practical network applications.

E. Parameter Sensitive

We investigate the sensitivity of our proposed model in this section. We conduct this experiment on three datasets and report the classification results when the training ratio is 30%.

Dimension size. The effect of the dimension size on classification performance is shown in Fig. 4(a). When the dimension size is small, more useful information can be incorporated and the performance becomes better. However, too large a value of dimension size will introduce noise and redundant information even leading to a worse result.



(a) dimension size d

(b) largest search hops K

Fig. 4. Classification on the three datasets with different embedding size and largest search hops.

Largest search hops. The effect of the largest search hops on classification performance is shown in Fig. 4(b). When the largest search hops is small, more useful information can be exploited and the performance also becomes better. However, the farther the distance between two nodes and the less similar their attributes are. Too large a value of the largest search hops will introduce attribute noise, which leads to a worse performance. Actually, AANI can relief the impact of attribute noise through weighing the *k-th* order smoothed attribute adaptively. However, as the largest search hops keeps growing, more useless attribute information of neighborhood will be exploited leading to a worse result. Furthermore, too large the value of largest search hops will lead to an increase in time complexity.

VI. CONCLUSIONS

In this paper, we propose a sparse attributed embedding method named AANI. AANI significantly improves the performance of node representations through exploiting the information of current node and its neighborhood in a smoothing way. In addition, we introduce attention mechanism to adaptively assign weights for the smoothed information according to their respective importance on the node representation. Finally, we perform truncated random walks on the network and maximize the occurrence probability of current node and its context nodes. Extensive experiments have demonstrated that AANI outperforms the state-of-the-art embedding approaches. As for future work, we plan to extend our proposed model to perform network embedding in dynamic attributed networks.

References

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

[2] 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.

[3] P. Cui, X, Wang, J. Pei, and W. Zhou, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, pp. 833–852, 2018.

[4] S. T. Roweis, and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[5] J. B. Tenenbaum, D. S. Vin, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[6] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, pages 3111–3119, 2013.

[7] 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.

[8] A. Grover, and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Datg*, 2016, pp. 855–864.

[9] 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.

[10] 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.

[11] 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.

[12] X. Wang, L. Yu, K. Ren, G. Tao, W. Zhang, Y. Yu, et al, "Dynamic attention deep model for article recommendation by learning human editors' demonstration," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 2051–2059.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[14] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proceedings of the 24th International Joint Conference on Artificial Intelligence,* 2015, pp. 2111–2117.

[15] X. Huang, J. Li, and X. Hu. "Accelerated attributed network embedding," in *SIAM*, 2017, pp. 633–641.

[16] L. Liao, X. He, H. Zhang, and T. S. Chua. "Attributed social network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 12, pp. 2257–2270, 2018.

[17] 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.

[18] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *Proceedings of the 20th ACM International Conference on Web Search and Data Mining*, 2017, pp. 731–739.

[19] J. Liang, P. Jacobs, J. Sun, and S. Parthasarathy, "Semi-supervised embedding in attributed networks with outliers," in *Proceedings of the 2018 SIAM International Conference on Data Mining*, 2018, pp. 153–161.

[20] Z. Zhang, H. Yang, and J. Bu, et al, "ANRL: Attributed network representation learning via deep neural networks," in *IJCAI*, 2018, pp. 3155–3161.

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

[22] L. Du, Y. Wang, G. Song, Z. Lu and J. Wang, "Dynamic network embedding: An extended approach for skip-gram based network embedding," in *IJCAI*, 2018, pp. 2086–2092.

[23] S. Kumar, X. Zhang, and J. Leskovec, "Predicting dynamic embedding trajectory in temporal interaction networks," in *SIGKDD*, 2019, pp. 1296–1278.

[24] H. Li, M. R. Min, Y. Ge, and A. Kadav, "A context-aware attention network for interactive question answering," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 927–935.

[25] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, "Image captioning with semantic attention," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4651–4659.

[26] H. Wang, E. Chen, and Q. Liu, et al, "A united approach to learning sparse attributed network embedding," in *IEEE International Conference on Data Mining*, 2018, pp. 557–566.

[27] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

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

[30] 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.