NANE: Attributed Network Embedding with Local and Global Information

Jingjie Mo[1,2,3], Neng Gao[2,3(✉)], Yujing Zhou[1,2,3], Yang Pei[1,2,3], and Jiong Wang[1,2,3]

[1] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{mojingjie,zhouyujing,peiyang2,wangjiong}@iie.ac.cn
[2] State Key Laboratory of Information Security, Chinese Academy of Sciences, Beijing, China
gaoneng@iie.ac.cn
[3] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

Abstract. Attributed network embedding, which aims to map structural and attribute information into a latent vector space jointly, has attracted a surge of research attention in recent years. However, existing methods mostly concentrate on either the local proximity (i.e., the pairwise similarity of connected nodes) or the global proximity (e.g., the similarity of nodes' correlation in a global perspective). How to learn the global and local information in structure and attribute into a same latent space simultaneously is an open yet challenging problem. To this end, we propose a **N**eural-based **A**ttributed **N**etwork **E**mbedding (**NANE**) approach. Firstly, an affinity matrix and an adjacency matrix are introduced to encode the attribute and structural information in terms of the overall picture separately. Then, we impose a neural-based framework with a pairwise constraint to learn the vector representation for each node. Specifically, an explicit loss function is designed to preserve the local and global similarity jointly. Empirically, we evaluate the performance of **NANE** through node classification and clustering tasks on three real-world datasets. Our method achieves significant performance compared with state-of-the-art baselines.

Keywords: Attributed social networks · Deep learning
Local and global information · Pairwise constraint

1 Introduction

Social networks are ubiquitous in our daily lives, ranging from online social-networking sites such as Facebook and Weibo, biological gene-disease networks [6,18], to citation networks [14,16]. Network embedding, which maps the information of each node into a latent space, has grown up to be an effective method in social network data mining. As a result, various data mining methods, e.g.,

H. Hacid et al. (Eds.): WISE 2018, LNCS 11233, pp. 247–261, 2018.
https://doi.org/10.1007/978-3-030-02922-7_17

group clustering [27], link prediction [21], anomaly detection [3], node classification [2,21], can be directly conducted in the latent space.

Early works(e.g., DeepWalk [24], LINE [28], node2vec [9]) explore the effect of local and global structure similarity [8,12,17,31] on network embedding. Local structure similarity indicates that a pair of nodes with edges are more prone to be similar. On the other hand, global structure similarity reveals node status in the network. For instance, a pair of nodes with semblable structural neighbors tend to be closer. However, these methods need to be improved for lack of abundant nodes' attribute information. Recently, several algorithms pay attention to attributed network embedding leveraging structural and attribute information both. LANE [10] only takes the local attribute similarity into consideration which means that nodes in the network with similar attribute are more likely to be in a same community. UPP-SNE [35] is more prone to capturing global structural and attribute similarity under the framework of DeepWalk. How to extract the local and global information in structure and attribute jointly is still an arduous problem.

In this paper, we propose a **N**eural-based **A**ttributed **N**etwork **E**mbedding framework named **NANE** to address the aforementioned problem. In our approach, various information is comprehensively considered, including the local and global information in structure and attribute. We impose an autoencoder model to encode the global attribute and structural information into a latent space with a pairwise constraint to preserve the local information. Specifically, we regard the attribute similarity of two nodes as an uncertain link and the similarity indicates the possibility of connecting two nodes. On top of that, an affinity matrix is introduced to represent the attribute global proximity.

In conclusion, the main contributions can be summarized as follows:

- We propose a generic neural-based framework **NANE** to represent attributed social network capturing both structural and attribute non-linear similarity. To our best of knowledge, our model is the first attempt to capture the local and global similarity in structure and attribute jointly. Specifically, we introduce an affinity matrix to measure the global attribute similarity among nodes.
- We empirically conduct experiments on three real-world datasets with multi-class classification of vertices and node clustering tasks. Comparing with cutting-edge network embedding algorithms, we evaluate the effectiveness of **NANE**.

The rest of this paper is organized as follows. Firstly, we discuss the related work in Sect. 2, and then we show some definitions in our work in Sect. 3. On top of that, we propose the **NANE** algorithm in Sect. 4, followed by experiments in Sect. 5. Finally, Sect. 6 concludes the paper and visions the future work.

2 Related Work

In this section, we briefly summarize the development of network embedding methods which we can simply divide into two parts. The first part is named

structure-based network embedding which focuses on structural information only. The latter is named content-aware network embedding that combines additional information with network structure for a better performance on graph representation.

2.1 Structure-Based Network Embedding

Some earlier works utilize manifold learning to capture structure proximity [1, 25, 29]. Recently, inspired by Skip-Gram [19, 20], DeepWalk [24] imposes random walks to generate sequences of nodes, and feeds them into Skip-Gram to capture the similarity of nodes. However, DeepWalk is prone to preserving the global structural proximity since the limit of random walks. On top of that, node2vec [9] introduces biased random walks to balance the global and local structural information. LINE [28] designs an explicit objective function to capture both global and local structural information while it lacks further fusion.

On the other hand, some researchers try to introduce neural networks to extract node high non-linearities. DNGR [5] adopts a random surfing model to capture network structure and reduces it into a limit dimension by a SDAE model. It happens that there is a similar case. SDNE [33] also exploits a deep neural network with a supervised component to extract the structural information.

All the methods mentioned above are based on network structure only without considering attribute information which is extremely beneficial to graph representation.

2.2 Content-Aware Network Embedding

Some recent efforts have been devoted to leveraging additional information for a better performance.

TADW [34], a text-associated DeepWalk model, creatively explores the contribution of nodes contents in network embedding. It imposes matrix factorization to encode text features into network representation learning. TriDNR [22] exploits DeepWalk and Doc2Vec [13] to extract structural information and capture content information respectively. Relying on a late fusion which is a series of weighted sums only, it ignores the correlation between structure and contents since it lacks further convergence. Further, these two models which concern about node contents only cannot handle noisy attribute information.

LANE [10] is the first attempt to utilize label information and node attributes. It utilizes three matrices to measure networks: the network adjacent matrix, node content-level similarity matrix and node label-level similarity matrix, and then maps them into the same vector space by dimension reduction. LANE projects node embedding with matrix factorization which cannot reflect the non-linear correlation between nodes. UPP-SNE [35] is the first attempt to take user profiles into consideration. The basic idea is to extract the similarity of nodes by random walks and embed user profile information via a non-linear mapping into a low-dimensional latent space. However, UPP-SNE is more prone to extracting the global proximity while it ignores the local information.

3 Problem Definition

We first summarize some notations used in this paper. We denote scalars as lowercase alphabets(e.g., n) and represent vectors as boldface lowercase alphabets(e.g., $\mathbf{s}$). Moreover, matrices are represented by boldface uppercase alphabets(e.g., $\mathbf{S}$). $\mathbf{s}_i$ is the i^{th} row of a matrix $\mathbf{S}$, and the $(i,j)^{th}$ element of it is denoted by s_{ij}. We list the main notations in Table 1.

We regard a social network as a homogeneous attributed network which indicates that there is only one relationship between nodes, each edge is undirected and some nodes have their attributes. Under these assumptions, we denote a social graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{U})$, where $\mathcal{V}$ is a set of n nodes, $\mathcal{E}$ is a set of edges and $\mathbf{U}$ is the node attribute matrix. $\mathbf{A}$ is the adjacent matrix, and a_{ij} is defined as 1 if there is a edge between node i and node j.

The aim of this paper is to project the nodes into a low-dimensional vector space while preserving structural and attribute information jointly. It is vital to map the matrices $\mathbf{U}$,$\mathbf{A}$ into the same latent space jointly. In this end, we propose a neural-based attributed network embedding model to learn a comprehensive representation.

Table 1. Main notations and descriptions

Notations	Descriptions
n	Total number of nodes in the network
m	Total number of attribute categories of all nodes
d	The dimension of embedding space
t	The dimension of the output of self-feedforward layer
K	The number of deep autoencoder layers
$\mathbf{A} \in \mathbb{R}^{n \times n}$	The adjacent matrix
$\mathbf{U} \in \mathbb{R}^{n \times m}$	The user attribute information matrix
$\mathbf{S} \in \mathbb{R}^{n \times n}$	The node attribute similarity matrix
$\mathbf{X} \in \mathbb{R}^{n \times d}$	The final embedding matrix
$\widehat{\mathbf{R}} \in \mathbb{R}^{n \times 2n}$	The output of reconstruction layer
$\mathbf{h}_i^{(j)}$	j^{th} Hidden layer of node i
$\mathbf{W}^{(k)}$,$\mathbf{b}^{(k)}$	The k^{th} hidden layer weight matrix and biases
$\mathbf{W}_{struc}$,$\mathbf{W}_{attr}$	The weight matrix of structural information and attribute information

4 Methodology

In this section, we first elaborate the framework of our final model **NANE** preserving structural and attribute information jointly. And then we give the explicit loss function and its optimization in detail.

4.1 Overall Architecture

In this paper, we propose a neural-based attributed network embedding model to capture structural and attribute information jointly. As shown in Fig. 1, **NANE** consists of three major steps. Specifically, we first convert structural and attribute information to a set of binary features, and then leverage a self-feedforward layer to measure the weight of each feature. We then concatenate the output of self-feedforward layer and feed it into a neural network structure. The early fusion operation makes it possible to optimize all parameters simultaneously. To capture the highly non-linear node correlation, we utilize a deep autoencoder [26] model to reconstruct overall information and design a specific loss function to preserve both local and global similarity in the end. The details of each step are elaborated as follows.

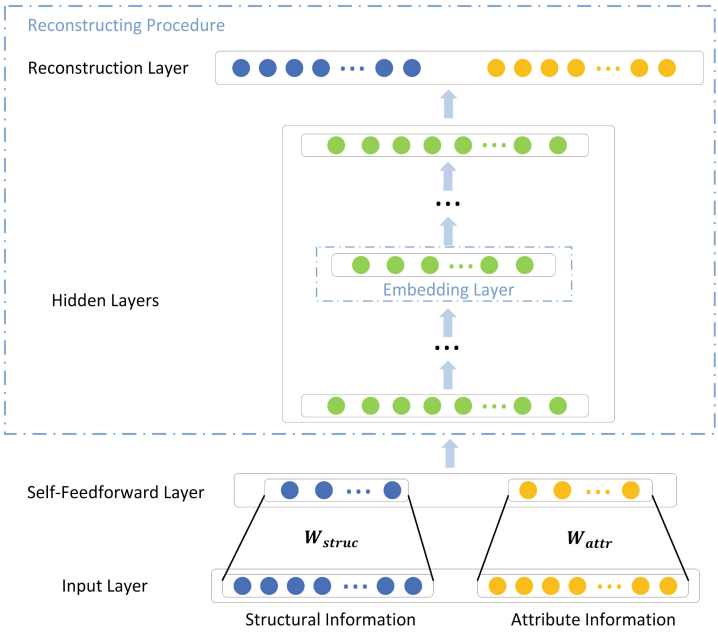


Fig. 1. The framework of **NANE**

4.2 Global Information Encoding

To economize the global attribute and structural information, we introduce an affinity matrix **S** and an adjacent matrix **A** respectively. In this paper, we take a row of the adjacency matrix to represent for structural information. For example, $\boldsymbol{a}_i$ is the structural information of vertex i. Furthermore, we utilize a cosine similarity to represent the global attribute information. We simply take a row of the

similarity matrix $\mathbf{S}$ to represent the attribute information, and $\boldsymbol{s}_i$ is the attribute similarity information of vertex i.

To generate the attribute affinity matrix $\mathbf{S}$, we need to define the user attribute information matrix $\mathbf{U}$ in the first place. As is well-known, attribute information is rich and diverse in social networks. However, it is inevitably incomplete and noisy due to its heterogeneity and feature of manual filling. To tackle this problem, we first convert all discrete attributes, e.g., user demographics [4], user interest [23], to a set of binary features by one-hot encoding. For instance, the marital status attribute has four values $\{married, single, divorced, widowed\}$, we can encode a married user as the vector $\mathbf{v} = \{1, 0, 0, 0\}$, where the first binary feature of value 1 represents married. For continuous attributes, e.g. age, we normalize it to reduce the impact of value by Max-Min Normalization. For missing attribute values, we set the feature vector to all zeros. Thus, we aggregate all the feature vectors together and then we obtain the user attribute information matrix $\mathbf{U}$. We obtain s_{ij} by calculating the cosine similarity of $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$.

4.3 Self-Feedforward Layer

To weight features in structure and attribute, we design a self-feedforward layer for both structure and attribute embedding. As shown in Fig. 1, the self-feedforward layer consists of two fully connected layers, which convert features in structural and attribute information into two dense vectors respectively. And we apply dropout, residual connections and layer normalization on the self-feedforward layer to prevent overfitting and improve the efficiency of neural network. The weights of each feature are denoted as $\mathbf{W}_{struc}$ and $\mathbf{W}_{attr}$ separately. The final output of self-feedforward layer of vertex i is denoted as $\mathbf{h}_i^{(0)}$, which can be expressed as follows:

$$\mathbf{h}_i^{(0)} = \left[Dropout(\sigma(\mathbf{W}_{struc} \cdot \mathbf{a}_i)), \lambda Dropout(\sigma(\mathbf{W}_{attr} \cdot \mathbf{s}_i))\right] \tag{1}$$

where $\mathbf{h}_i^{(0)}$ denotes the input layer of the autoencoder of vertex i, λ adjusts the impact of attributes, σ denotes the activation function, $\mathbf{W}_{struc}$ and $\mathbf{W}_{attr}$ are parameters we need to learn, which measure the weights of structural and attribute information respectively. For a better convergence and preventing overfitting, we design an $\mathcal{L}2$-norm regularizer in this part, which is defined as follows:

$$\mathcal{L}_{F-reg} = \|\mathbf{W}_{struc}\|_F^2 + \|\mathbf{W}_{attr}\|_F^2 \tag{2}$$

where $\|\cdot\|_F^2$ denotes the square of F-norm.

4.4 Reconstructing Procedure

To capture the local and global proximity in structure and attribute jointly, we extend a deep autoencoder model to measure the interplay between structure and attribute and encode them into a latent space. A deep autoencoder

consists of two parts, i.e. the encoder and decoder. At the encode step, multilayer non-linear function $f(\cdot)$ is applied to map the input data into a low-dimensional vector space, which is also called embedding space. On the contrary, there are also multilayer mirrored non-linear function $g(\cdot)$ is extended to map the embedding space to the reconstruction space. In this paper, the input data of deep autoencoder is the output of self-feedforward layer $\mathbf{h}_i^{(0)}$, and the representation of hidden layers can be denoted as follows:

$$\mathbf{h}_i^{(k+1)} = \sigma(\mathbf{W}^{(k)} \cdot \mathbf{h}_i^{(k)} + \mathbf{b}^{(k)}) \tag{3}$$

where $W^{(k)}$, $b^{(k)}$ is the k^{th} hidden layer weight matrix and biases.

The output of reconstruction layer is denoted as $\widehat{\mathbf{R}}$. Our goal of this step is to minimize the reconstruction error of output and input. The input data $\mathbf{R}$ that need to be reconstructed is shown as follows:

$$\mathbf{R} = \left[\mathbf{A}, \lambda \mathbf{S}\right] \tag{4}$$

And the loss function is denoted as follows:

$$\mathcal{L} = \left\| \widehat{\mathbf{R}} - \mathbf{R} \right\|_F^2 \tag{5}$$

However, due to the sparsity of the input data, the number of zero elements in $\mathbf{R}$ is much larger than that of non-zero ones, which means that it is more prone to reconstruct the zero elements rather than non-zero ones. However, it is contrary to our intention. To address this problem, we impose an offset coefficient matrix to reset the weights of different elements, and the redesigned loss function is denoted as follows:

$$\mathcal{L}_{global} = \left\| (\widehat{\mathbf{R}} - \mathbf{R}) \odot \mathbf{B} \right\|_F^2 \tag{6}$$

where $\mathbf{B}$ is the offset coefficient vector, b_{ij}=1 when r_{ij}=0 while $\mathrm{b}_{ij} = \beta > 1$ when r_{ij}=1, and $\odot$ means the Hadamard product. With the help of the deep autoencoder, the vectors of the vertexes which have semblable features would have similar representation. However, it is not only necessary to capture the global similarity, but also vital to preserve the local similarity. A pair of nodes with edges should also have semblable embedding i.e. the local structure similarity. Therefore, we aggrandize a loss function for local structure similarity, and the expression is shown as follows:

$$\mathcal{L}_{struc-local} = \sum_{i,j=1}^{n} a_{ij} \left\| \mathbf{x}_i - \mathbf{x}_j \right\|_2^2 \tag{7}$$

where a_{ij} is an element of adjacency matrix, $a_{ij} = 1$ when there is a link between node i and node j. x_i denotes the final embedding of node i. The local structure loss function aims to make the embedding of two connected nodes closer.

Similar with the local structure similarity, we also design a loss function to constraint the local attribute similarity. A pair of nodes with semblable attributes should also have semblable embedding. The loss function for local attribute similarity is denoted as follows:

$$\mathcal{L}_{attr-local} = \sum_{i,j=1}^{n} s_{ij} \left\| \mathbf{x}_i - \mathbf{x}_j \right\|_2^2 \tag{8}$$

where s_{ij} is an element of attribute similarity matrix. The greater value of s_{ij}, the attributes of node i and node j are more similar. Akin to the local structure loss function, the local attribute loss function aims to make the embedding of two nodes which have similar attributes closer.

4.5 Loss Functions and Optimization

To preserve the global and local similarity simultaneously, we combine the aforementioned loss functions in a integrated framework and propose a unified structure and attribute preserving model **NANE**. The joint loss function is denoted as follows:

$$\begin{aligned} \mathcal{L} &= \alpha \mathcal{L}_{global} + \gamma \mathcal{L}_{struc-local} + \theta \mathcal{L}_{attr-local} + \eta \mathcal{L}_{reg} \\ &= \alpha \left\| (\widehat{\mathbf{R}} - \mathbf{R}) \odot \mathbf{B} \right\|_F^2 + \gamma \sum_{i,j=1}^{n} a_{ij} \left\| \mathbf{x}_i - \mathbf{x}_j \right\|_2^2 \\ &+ \theta \sum_{i,j=1}^{n} s_{ij} \left\| \mathbf{x}_i - \mathbf{x}_j \right\|_2^2 + \eta \mathcal{L}_{reg} \end{aligned} \tag{9}$$

where α,γ,θ,η are four hyper-parameters to adjust the weights of each part. Moreover, $\mathcal{L}_{reg}$ is the total regularization which consists of two parts: the regularization of self-feedforward layer $\mathcal{L}_{F-reg}$ and the regularization of deep autoencoder $\mathcal{L}_{ae-reg}$. We define $\mathcal{L}_{reg}$ as

$$\begin{aligned} \mathcal{L}_{reg} &= \mathcal{L}_{F-reg} + \mathcal{L}_{ae-reg} \\ &= \left\| \mathbf{W}_{struc} \right\|_F^2 + \left\| \mathbf{W}_{attr} \right\|_F^2 + \sum_{k=1}^{K} \left(\left\| \mathbf{W}^{(k)} \right\|_F^2 + \left\| \mathbf{b}^{(k)} \right\|_F^2 \right) \end{aligned} \tag{10}$$

To optimize the aforementioned framework, we apply RMSProp to minimize the objective in Eq.(9), which is able to adjust the learning rate for each parameter. We feed mini-batch into our model each time to accelerate the speed of training. Besides, in order to prevent falling into a local optimum solution, it is essential to find a good set of initialized parameters. Therefore, we adapt Restricted Boltzmann Machine to pre-train the parameters at first, which is a classic method of parameter initialization in neural network [7]. The integrated algorithm is presented in Algorithm 1.

Algorithm 1. NANE model

Input: The network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{U})$ with the adjacency matrix $\mathbf{A}$ and hyper-parameters
Output: Network embedding $\mathbf{X}$

1: Extract the feature vectors and calculate the attribute similarity matrix $\mathbf{S}$;
2: Feed matrix **A,S** into self-feedforward layer, and merge them together;
3: Pre-train the parameters of deep autoencoder through RBM to get initial parameter values;
4: Feed the output of self-feedforward layer into deep autoencoder;
5: **repeat**
6: Based on the input data and the weights of each layer, apply Eq.(1) to generate the reconstruct matrix $\widehat{\mathbf{R}}$;
7: minimize Eq. (9) by RMSProp, and update parameters at each epoch;
8: **until** converge
9: Acquire the network embedding $\mathbf{X} = \mathbf{H}^{(K/2)}$

5 Experiments

In this section, we evaluate our method by performing experiments on three real-world network datasets and compare it with several state-of-the-art baseline algorithms.

5.1 Datasets

We conduct experiments on three real-world networks: Facebook, Hamilton and Rochester. The statistics of the three datasets is summarized in Table 2.

Ego-Facebook is an ego-network which was collected from survey participants using the Facebook app. The dataset contains 1403-dimensional node features, 4039 nodes and 88234 edges. Besides, people education type is used as group label [15].

Hamilton and **Rochester** are two datasets collected by Adam D'Angelo of Facebook, consists of nodes from the Facebook networks at each of 100 American institutions [30]. Each node contains 7 attributes: student/faculty status flag, gender, major, second major/minor, dorm/house, year, and high school, which is described by a 144-dimensional and 235-dimensional feature vector respectively, and student/faculty status flag is selected as group class. Two datasets include 2314 nodes, 192788 edges and 4563 nodes, 322808 edges separately.

Table 2. The statistics of the three datasets

Dataset	Node	Edge	Attribute
Facebook	4039	88234	1403
Hamilton	2314	192788	144
Rochester	4563	322808	235

5.2 Baseline Methods

We compare **NANE** with several state-of-the-art network embedding methods, which are divided into two categories.

Structure-Based NRL Methods

- DeepWalk [24] generates node sequences by random walks, and feed them into Skip-Gram model to learn network embedding.
- node2vec [9] introduces biased random walks to DeepWalk, which aims to capture the local and global structure jointly.
- LINE [28] imposes two separate objective functions to extract the first-order and the second-order proximity both.
- SDNE [33] leverages a deep neural network which is a non-linear mapping operation to exploit structural information.

Attribute-Aware NRL Methods

- LANE [10] fuses structural, attribute and label information together to preserve node similarity. Here, we only use the version without utilizing label information.
- UPP-SNE [35] generates random walks to capture node pairwise similarity and embeds user profile information into a low-dimensional vector space.

5.3 Parameter Settings

For a fair comparison, we set the embedding dimension to 256 for all methods. In DeepWalk, node2vec and UPP-SNE, we set the window size t to 10, the walk length l to 80. In node2vec, we empirically set the return hyperparameter p to 2.0, and the in-out hyperparameter q to 0.5. In LINE, we set the first-order vector dimension to 128, and the second-order vector dimension to 128 in the same way. In SDNE, we set the weight γ, α, β to 1, 500, 100 respectively, and the learning rate is 0.01. In LANE, we tune the hyper-parameters of β_1, β_2 by grid search. In UPP-SNE, we set the number of iterations to 20. For our method, we use a grid search to find the best parameters. We set α to 1000, β to 100, and γ,θ and η to 1. The rest parameters are given in Table 3.

Table 3. Parameter settings of the three datasets

Dataset	Batch size	λ	t
Facebook	600	1.5	256
Hamilton	400	1.5	1000
Rochester	600	1	256

5.4 Node Classification

We first evaluate the effectiveness of **NANE** by multi-class node classification. To be fair, we range the training size from 15% to 75% by taking 10% as a step, and apply a rbf-kernel svm classifier with $\gamma = 100$ to all of the generated node representations. For each training size, we split train and test set in a random way for 10-times, and then conduct 5-fold cross-validation and output the average micro F1-score of node classification on three different datasets.

Table 4. Node classification F1-score(%) on Facebook

Train size	15%	25%	35%	45%	55%	65%	75%
LINE	68.93	68.93	68.56	68.87	69.73	68.78	68.56
DeepWalk	68.31	68.98	69.76	69.98	69.14	68.88	69.01
node2vec	68.84	69.34	69.5	69.89	69.36	69.34	69.01
SDNE	63.51	63.73	64.09	64.32	63.32	64.01	63.13
LANE	70.03	70.2	69.9	70.2	69.86	69.66	70.1
UPP-SNE	85.53	85.87	86.41	86.81	87.18	87.13	87.82
NANE	**88.62**	**88.92**	**89.16**	**89.18**	**89.03**	**89.79**	**90.97**

Table 5. Node classification F1-score(%) on Hamilton

Train size	15%	25%	35%	45%	55%	65%	75%
LINE	90.04	90.45	91.18	91	91.41	91.56	92.36
DeepWalk	92.32	92.86	92.62	92.77	93.19	92.72	92.75
node2vec	92.43	92.86	92.56	92.46	92.9	93.09	93.09
SDNE	91.2	91.65	92.36	92.46	92.71	92.59	92.4
LANE	79.36	79.26	79.27	82.88	87.04	89.88	92.26
UPP-SNE	93.86	93.55	93.52	93.17	93.75	93.45	94.25
NANE	**94.76**	**94.47**	**94.35**	**94.27**	**94.32**	**94.44**	**95.16**

Tables 4, 5 and 6 show the average classification accuracy of all the methods on Facebook, Hamilton and Rochester, where the best results are bold-faced. It is not hard to see that our method consistently yields the best classification results among all the baselines. Extraordinary accuracy compared with structure-based NRL algorithms demonstrates that our method works well on the fusion of attribute information. Especially on Facebook, **NANE** achieves more than 30% improvement over all the structure-based NRL baselines, pointing to the significant performance on node classification. Moreover, **NANE** also outperforms all the attribute-aware NRL baselines, demonstrating its effectiveness on capturing the local and global similarity in structure and attribute. Our method imposes an early fusion of structural and attribute information, which

Table 6. Node classification F1-score(%) on Rochester

Train size	15%	25%	35%	45%	55%	65%	75%
LINE	86.22	86.58	86.73	87.28	87.79	87.65	87.56
DeepWalk	88.1	88.29	88.57	88.97	88.9	89.74	88.69
node2vec	88.32	88.14	88.71	89.4	89.29	89.36	88.52
SDNE	86.08	86.21	86.92	86.57	87.2	86.73	85.89
LANE	80.97	81.27	81.56	81.35	80.87	81.1	80.89
UPP-SNE	89.46	89.89	90.26	90.08	89.87	89.80	88.96
NANE	**89.53**	**90.61**	**90.86**	**90.58**	**90.8**	**90.61**	**90.62**

enables our method to exploit the interplay between structure and attribute. On top of that, our method economizes the global information with a pairwise constraint, resulting in remarkable consequences.

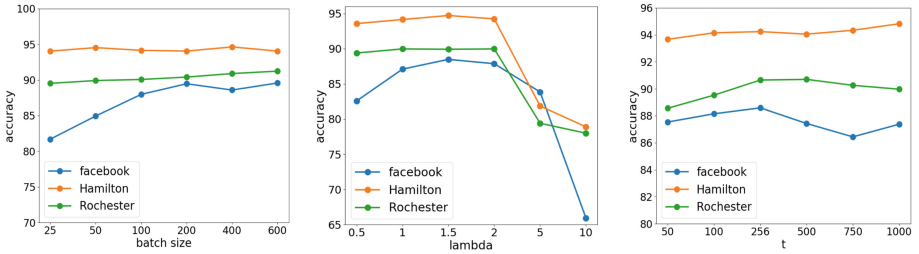
5.5 Parameter Sensitivity

In this section, we explore the parameter sensitivity of our method. We select three crucial parameter batch size, λ and t to conduct our experiments and investigate how these parameters affect the results. The train ratio is selected as 55%. We report the accuracy of node classification on different datasets with disparate parameters. In turns, we study the effect of one parameter with fixing the rest. And the results of our experiments are shown in Fig. 2.

We first show how the batch size affects the performance in Fig. 2(a). As we can see, the performance of our method on Hamilton and Rochester is stable with different values. However, performance on facebook is much fluctuant. The best value of batch size on facebook is 600. We then show how the weight of attribute information λ influence the result in Fig. 2(b). It not hard to see that when the weight of attribute information approach to 1, which means structural information is as important as attribute information, the performance is much better. However, the performance of our method falls badly when *lambda* is larger than 10. It indicates that it may leads to the very opposite effect when the influence of attributes is too large. At last, how the dimension of the output of self-feedforward layer t affects the result is shown in Fig. 2(c). We can see that when t is close to the embedding dimension, the performance is much better in most case.

5.6 Node Clustering

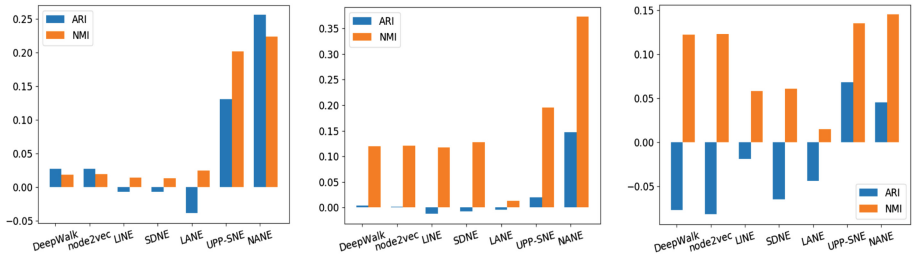
For node clustering task, we apply K-Means++ to network embedding on three datasets and leverage ARI [11] and NMI [32] to evaluate the consequence of clustering. We set the number of clusters same as the group number, calculate the indexes 10 times to reduce occasionality. the average results are shown in Fig. 3.



(a) batch size (b) λ (c) t

Fig. 2. The impact of parameters batch size, λ, and t

As we can see, **NANE** achieves the best performance among almost all the cutting-edge baselines. Similar clusterings have a positive ARI, whereas negative values indicate poor performance. Furthermore, values of exactly 0 on NMI present purely independent label assignments. On Facebook, the values of ARI are lower than zero for LINE and SDNE, verifying that the representations of LINE and SDNE have independent labelings. Specifically, **NANE** achieves 0.125 improvement compared with the second best results. On Hamilton, **NANE** also yields the best clustering results, outperforming 0.127 and 0.178 lift on ARI and NMI respectively. It firmly demonstrates the better performance of **NANE** on unsupervised learning task.



(a) Facebook (b) Hamilton (c) Rochester

Fig. 3. The results of node clustering on three datasets

6 Conclusion and Future Work

Attributed network, due to its inherent heterogeneity and sparsity, presents new challenges for many tasks. In this paper, we propose a Neural-based Attributed Network Embedding method to capture structural and attribute information jointly. We deem that it is crucial to explore the global and local similarity simultaneously for a better representation. Firstly, we impose a fully-connected layer with some dropout to extract the relevance in attribute and structure respectively. And then we leverage a deep autoencoder model which is a

non-linear mapping to reconstruct the global information with a local constraint. Experiments on three different real-world datasets demonstrate the remarkable performance of our method comparing with cutting-edge algorithms. Our future work in this area will focus on the following directions. We will develop **NANE** to a multi-task learning algorithm and make it suitable for large-scale industrial-grade data. Furthermore, We are also interested in exploring how to capture user behaviors in time series and user images and map them into embedding space.

Acknowledgements. This work is partially supported by National Natural Science Foundation of China (No.U163620068) and National Key Research and Development Program of China.

References

1. Belkin, M., Niyogi, P.: Laplacian Eigenmaps and spectral techniques for embedding and clustering. In: International Conference on Neural Information Processing Systems: Natural and Synthetic, pp. 585–591 (2002)
2. Bhagat, S., Cormode, G., Muthukrishnan, S.: Node Classification in Social Networks. Springer, New York (2011)
3. Bhuyan, M.H., Bhattacharyya, D.K., Kalita, J.K.: Network anomaly detection: methods, systems and tools. IEEE Commun. Surv. Tutor. **16**(1), 303–336 (2014)
4. Burger, J.D., Henderson, J., Kim, G., Zarrella, G.: Discriminating gender on twitter. In: Conference on Empirical Methods in Natural Language Processing, pp. 1301–1309 (2011)
5. Cao, S., Lu, W., Xu, Q.: Deep neural networks for learning graph representations. In: Thirtieth AAAI Conference on Artificial Intelligence, pp. 1145–1152 (2016)
6. Dong, Y., Zhang, J., Tang, J., Chawla, N.V., Wang, B.: Coupledlp: link prediction in coupled networks. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 199–208 (2015)
7. Erhan, D., Bengio, Y., Courville, A., Manzagol, P.A., Vincent, P., Bengio, S.: Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. **11**(3), 625–660 (2010)
8. Granovetter, M.: The strength of weak ties: a network theory revisited. Sociol. Theory **1**(6), 201–233 (1983)
9. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: KDD 2016, pp. 855–864 (2016)
10. Huang, X., Li, J., Hu, X.: Label informed attributed network embedding. In: Tenth ACM International Conference on Web Search and Data Mining, pp. 731–739 (2017)
11. Hubert, L., Arabie, P.: Comparing partitions. J. Classification **2**(1), 193–218 (1985)
12. Kossinets, G., Watts, D.J.: Origins of homophily in an evolving social network1. Am. J. Sociol. **115**(2), 405–450 (2009)
13. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents, vol. 4, p. II-1188 (2014)
14. Leskovec, J.: Graphs over time: densification laws, shrinking diameters, explanations and realistic generators. In: KDD, pp. 177–187 (2005)
15. Mcauley, J., Leskovec, J.: Learning to discover social circles in ego networks. In: International Conference on Neural Information Processing Systems, pp. 539–547 (2012)

16. Mccallum, A.K., Nigam, K., Rennie, J., Seymore, K.: Automating the construction of internet portals with machine learning. Inf. Retr. **3**(2), 127–163 (2000)
17. Mcpherson, J.M., Smith-Lovin, L.: Homophily in voluntary organizations: status distance and the composition of face-to-face groups. Am. Sociol. Rev. **52**(3), 370–379 (1987)
18. Menche, J., et al.: Disease networks. Uncovering disease-disease relationships through the incomplete interactome. Science **347**(6224), 1257601 (2015)
19. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Computer Science (2013)
20. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality, vol. 26, pp. 3111–3119 (2013)
21. Narayanan, A., Chandramohan, M., Chen, L., Liu, Y., Saminathan, S.: subgraph2vec: learning distributed representations of rooted sub-graphs from large graphs (2016)
22. Pan, S., Wu, J., Zhu, X., Zhang, C., Wang, Y.: Tri-party deep network representation. In: International Joint Conference on Artificial Intelligence, pp. 1895–1901 (2016)
23. Pennacchiotti, M., Popescu, A.M.: Democrats, republicans and starbucks afficionados: user classification in twitter. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 430–438 (2011)
24. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710 (2014)
25. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–6 (2000)
26. Salakhutdinov, R., Hinton, G.: Semantic Hashing. Elsevier Science Inc., Amsterdam (2009)
27. Tang, J., Liu, J., Zhang, M., Mei, Q.: Visualizing large-scale and high-dimensional data, vol. 37(7), pp. 287–297 (2016)
28. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line:large-scale information network embedding, vol. 2, pp. 1067–1077 (2015)
29. Tenenbaum, J.B., Silva, V.D., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**(5500), 2319 (2000)
30. Traud, A.L., Mucha, P.J., Porter, M.A.: Social structure of facebook networks. Physica A Stat. Mech. Appl. **391**(16), 4165–4180 (2012)
31. Tsur, O., Rappoport, A.: What's in a hashtag? Content based prediction of the spread of ideas in microblogging communities. In: ACM International Conference on Web Search and Data Mining, pp. 643–652 (2012)
32. Vinh, N.X., Epps, J., Bailey, J.: Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance (2010). JMLR.org
33. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1225–1234 (2016)
34. Yang, C., Liu, Z., Zhao, D., Sun, M., Chang, E.Y.: Network representation learning with rich text information. In: International Conference on Artificial Intelligence, pp. 2111–2117 (2015)
35. Zhang, D., Yin, J., Zhu, X., Zhang, C.: User profile preserving social network embedding. In: Twenty-Sixth International Joint Conference on Artificial Intelligence, pp. 3378–3384 (2017)