NEGAN:Network Embedding based on Generative Adversarial Networks

Yinfeng Ban*†, Juhua Pu†* Yujun Chen*†, Yuanhong Wang*†
*State Key Laboratory of Software Development Environment, Beihang University, Beijing, China
†Research Institute of Beihang University in Shenzhen, Shenzhen, China
Email: banyinfeng@buaa.edu.cn, pujh@buaa.edu.cn, chenjohn@buaa.edu.cn, lucienwang@buaa.edu.cn

***Abstract*—Network embedding, also known as graph representation, is a classical topic in data mining. It has been widely used in real-world network applications such as node classification and community detection. However, it remains open to find a method that is scalable and preserves both structure and content information. Based on generative adversarial networks, we propose an unsupervised network embedding framework NEGAN, which is featured by combining graph topology and node content. In NEGAN, network nodes are mapped to the target space in a highly flexible non-linear way, guided by the content of the nodes. This mapping is learned from the generator of the generative adversarial networks, and node adjacency in the input network is preserved. Experiments on real datasets show that NEGAN outperforms all the existing methods on many scenarios including node classification, visualization and community detection tasks.**

***key words*—network embedding, generative adversarial networks, unsupervised representation learning**

I. Introduction

Recent years have witnessed increasing interest in studying real-world networks such as social networks [1], citation networks [2] and molecular networks [3]. A fundamental technique in handling real-world networks is network embedding, also known as graph representation, which embeds the networks into a low-dimension Euclidean space [4], [5]. It has been actively studied and widely used in numerous real applications.

However, some basic challenges in network embedding remain far from fully solved. First, the embedding should be both content-aware and structure-aware. The nodes in real-world networks are usually rich in content, and the content is often important to applications [6]. An ideal embedding should at least preserve the content similarity, in addition to structure information [7]. It is a challenge to find one embedding suitable for two disparate categories (namely, content and structure) [8]. Second, the embedding should be scalable since various real-world networks are huge in scale. Any method that uses too much parameters or is too time-consuming in computing does not work.

Big efforts have been made to solve the problems. Inspired by the Skip-Gram model in natural language processing, many approaches used word representation model to learn node representation from a corpus of random walks sequences generated from network [9], [10], [11]. These methods learn node representation that maximizes the co-occurrence probability of a target node and its context nodes. Though efficient in capturing structure information, these methods completely neglect the content of nodes.

In order to preserve both structure and content information, researchers have recently tried matrix factorization [12], [13], [14]. However, those methods are largely based on the non-parametric approach, where each network node is mapped to a free point in $\mathbb{R}^d$ (here d is the dimension of the target space). Therefore totally we have $O(nd)$ free parameters to learn, where n is the number of nodes. When n is large and d is even moderate, the number of parameters is formidable, not mention the computational hardness of matrix factorization itself. Thus, these methods are not scalable to large networks.

To solve the two challenges, we propose NEGAN, a generative adversarial network [15] architecture adapted for network embedding. Basically, we use LSTM to extract the content information of neighboring nodes, followed by employing GAN to model the relationship between a node and its neighbors. In this way, both structural proximity and node content similarity are preserved by the embedding. What'more, NEGAN is parametric and scalable. After training the deep neural network, the output of the generator is exactly the network embedding. We compare NEGAN with several state-of-the-art unsupervised graph embedding methods including DeepWalk [9], node2vec [10] and TADW [12]. The experiments are done on two real-world network datasets in three scenarios including node classification, community detection, and graph visualization. Results show that NEGAN outperforms all the other methods.

Based on generative adversarial networks (GAN), this paper aims at solving the two challenges simultaneously. Our main contributions are summarized as follows:

- We propose a generic framework NEGAN to learn network embedding by preserving both structural proximity and node content similarity.
- We propose a parametric non-linear method which could be used in large-scale network.
- Empirical evaluation indicates remarkable performance of NEGAN.

The rest of this paper is organized as follows. Section II discusses related works in network embedding. Section III presents the proposed NEGAN model. Section IV evaluates the performance of our method. Section V concludes our work and hints at future perspectives.

II. RELATED WORKS

A. Network Embedding

Some early works focused on embedding an existing network into a low-dimensional vector space for further analysis. DeepWalk [9] used random walks sampling strategy on network to generate node sequences. The generated node sequences were treated as sentences in language models [16] and maximize the co-occurrence probability of a target node and its context nodes to learn the embeddings. node2vec [10] modified the way of generating node sequences by balancing breadth-first sampling and depth-first sampling based on DeepWalk, and achieved better performance than DeepWalk. LINE [17] proposed clear objective functions, which can be used to preserve the first-order proximity and second-order proximity of nodes. SDNE [18] introduced deep models with multiple layers of non-linear functions to capture the highly non-linear network structure. However, all these methods only used the topology information to learn representation for graph nodes. In real-world networks, there exist a large amount of node content information in nodes. Structure-based methods obviously failed to capture such valuable information, thus it may result in less informative embeddings.

Some recent efforts have explored the possibility of integrating content and network topology information to learn better representations. TADW [12] showed that DeepWalk is equivalent to matrix factorization, and the author designed the factorized matrix to represent the probability that a node reaches another by random walks. TriDNR [19] respectively learned representaions from label-fused Doc2Vec model [20] and structure-based DeepWalk, then the two types of embeddings learned were linearly combined together.

All the above-mentioned approaches learn non-parametric representation for graph nodes, except SDNE [18], which differ from the parametric mapping learned by NEGAN in this paper. What's more, many of these unsupervised and semi-supervised learning approaches solve representation learning problem from only the topology information. TADW [12] requires matrix operation like SVD decomposition, which limits the method to deal with small scale networks. Moreover, the input of TADW is an approximate matrix. This will greatly compromise its embedding efficiency. TriDNR in [19] is a semi-supervised method and requires extra label information. To solve the problems in the existing methods, the unsupervised learning method NEGAN in this paper learns parametric representation considering node content feature and topology information.

B. Generative Adversarial Networks

Deep learning provides a promising solution to problems across research fields [21]. As a result, many typical Deep models have been considered for application in the field of network embedding. Convolutional Neural Networks was introduced in the process of learning embedding of heterogeneous network in [22]. Auto-encoder [23] was used to design a semi-supervised deep learning model for network embedding. However, these models focused mainly on learning network embedding from only the network topology information.

Generative Adversarial Networks (GAN), another architecture of deep neural networks, was first introduced in 2014 by Goodfellow [15] to generate real images. Since the successful application in unsupervised feature learning, GAN and its variants achieve excellent performance in more areas including information retrieval [24], domain adaption [25], sequence generation [26] and neural dialogue generation [27]. GAN designs a minimax strategy in the game theory by combining generative models and discriminative models. For our application, the GAN establishes a representation space that embedds nodes and their neighborhood nodes nearby as they have similar contents and short graph distances. The node content and node adjacency in network structure are tightly integrated by a unified cost function in GAN.

III. METHODOLOGY

In this section, we will discuss the details of NEGAN used for learning network embedding based on generative adversarial networks, then we formulate our method of representation learning from both network structure and node content information. At a systematic level, the framework shown in Fig. 1 is comprised of the following stages:

1) The sampling strategy of network modeling used for the input of the NEGAN.
2) The deep neural networks consists of two parts: LSTM and GAN.
3) The parametric method that we learned network embedding from NEGAN.

Real-world networks are more than links. In most cases, network nodes contain rich content information. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ be a given graph, where $\mathcal{V} = \{v_1, ..., v_n\}$ denotes the network nodes, $\mathcal{E} = \{e_{ij}\}_{i,j=1}^{\mathcal{V}}$ denotes the links between network nodes, and $\mathcal{X} \in \mathbb{R}^{d_x \times \mathcal{V}}$ is the corresponding node content feature matrix, where d_x is the number of feature for nodes. In this paper, network representation learning is to find an generator G, also can be understood as a mapping function, which projects content feature $\mathcal{X}_v$ of each node v in a graph as a vector $G(\mathcal{X}_v; \theta_G) \in \mathbb{R}^d$ in d-dimensional space. Here, θ_G is the parameter of the generator that will be learned. d is a pre-specified parameter denoting the number of dimensions of node embedding. The learned $G(\mathcal{X}_v; \theta_G)$ will be used in many important machine learning applications like node classification and community detection.

The recent methods like Deepwalk [9] and node2vec [10] formalize the network embedding problem as the optimization of an objective function that maximizes the co-occurrence log-probability of a target node and its neighbor nodes. Given a node v and its neighbor nodes $N_S(v)$, the representation for v, $G(v)$, is learned by maximizing the conditional probability:

$$\max_{G} \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{N}_S(v)} \log Pr(G(c) \,|\, G(v)) \tag{1}$$

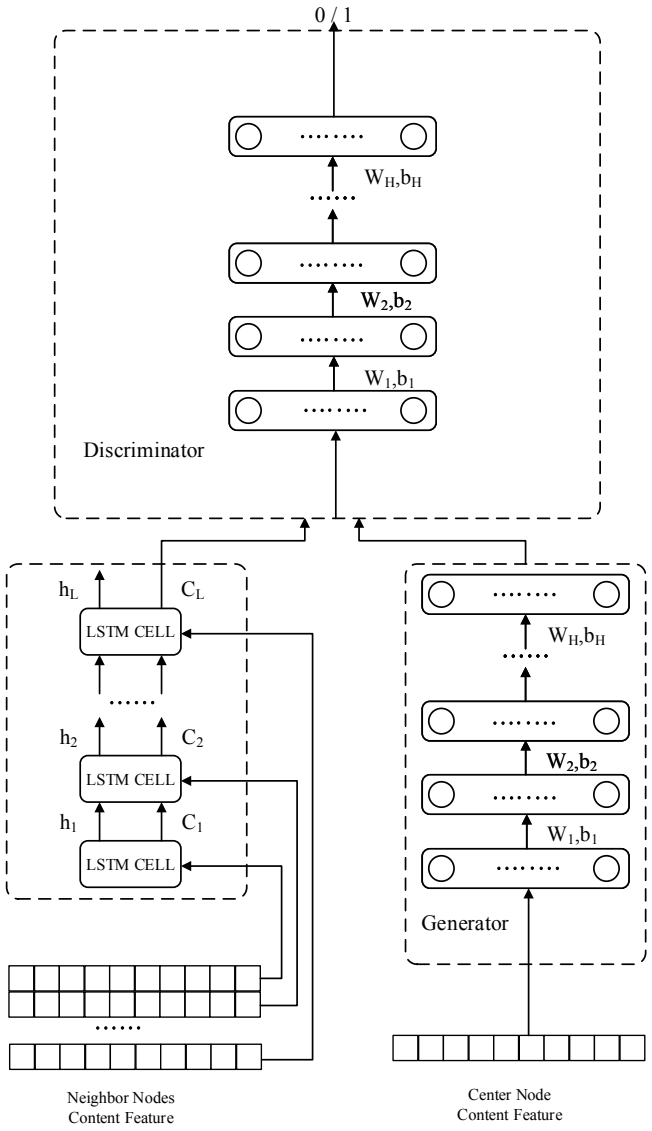


Fig. 1: The NEGAN architecture for parametric graph learning.

where for every target node $v \in \mathcal{V}$, $N_S(v) \subset \mathcal{V}$ is generated by a particular network sampling strategy S. Notice that the node representation $G(v)$ in node2vec is not parametrized.

The aim of our work is to learn a parametric mapping $G(\mathcal{X}_v; \theta_G)$. The non-linear function integrates the node content feature $\mathcal{X}$ and network structure, and the output is the parametric embedding we want. Following the formulation in Eq.(1), we maximize the log-probability of the co-ocurrance of neighbor nodes to node feature $\mathcal{X}$ in $N_S(v)$ by using parametric mapping function $G(\mathcal{X}_v; \theta_G)$:

$$\max_{\theta} \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{N}_S(v)} \log Pr(G(\mathcal{X}_c; \theta_G) \,|\, G(\mathcal{X}_v; \theta_G)) \qquad (2)$$

where $G_\theta(\mathcal{X}_v)$ is the parametric mapping function, same for all nodes V. The details of $G_\theta(\mathcal{X}_v)$ will be discussed in the following section III-B.

A. Sampling Strategy

In node2vec, the generalized sampling strategy for obtaining $N_S(v)$ of each v has shown its superiority over other methods like Deepwalk and LINE [17]. We thus follow the 2-nd order random walk algorithm in node2vec for obtaining the neighborhood nodes $N_S(v)$. We define a 2-nd order random walk with two parameters p and q which guide the 2-nd random walk: parameter p controls the likelihood of immediately revisiting a node in the walk, while parameter q allows the search to differentiate between inward and outward nodes. Since we learn representations for all nodes, we start to carry out the sampling strategy from every node. Neighborhood nodes feature $\mathcal{X}_N$ input to the GAN is regarded as real data distribution [15]. The start node feature $\mathcal{X}_v$ input to the generator of GAN is regarded as fake data distribution. Firstly, the start node S_1 of a walk sequence S is sampled uniformly from the node set $\mathcal{V}$. Afterward, a walk sequence $S = \{S_1, S_2, ..., S_j, ..., S_l\}$ is generated by using the 2-nd order random walk algorithm in node2vec, where S_j is the j-th node in S. So v is the staring node S_1, and $N_S(v) = \{S_2, ..., S_l\}$. There are two key hyper-parameters p and q that control the biased random walk. What's more, the hyper-parameters l and r control the walk length and walk times. The detailed algorithm is given in Algorithm 1.

Algorithm 1 Sampling pair (v, N_S) as training data

Input Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, parameter p, q,l,r,
Output pair set (v, N_S)

```
for all time t ∈ r do
    for all nodes v ∈ V do
        S = node2vecWalk(G, v, p, q, l)
        N_S = {S_2, .., S_l}
    end for
end for
Permute(v, N_S)
return (v, N_S)
```

B. Deep Network based on GAN

The main distinction of our work from the GAN described above is that our model designs a new deep neural network. Given the graph $\mathcal{G}$, we aim to learn the following three models:

Generator: $G(\mathcal{X}_v; \theta_G)$ tries to approximate the neighborhood nodes feature distribution and generates the node embedding $\mathcal{X}_v$ on the basis of its own node content. Network homophily states that similar nodes may be more likely to attach to each other than dissimilar ones based on node content. Thus, the aim of the generator in our method is to encode each node v to mimic the feature of the neighborhood nodes $\mathcal{X}_N$ to generate a new representation in target space.

Discriminator: $D(x; \theta_D)$ outputs a single scalar, which represents the probability that x comes from the neighborhood nodes feature rather than the output of generator $G(\mathcal{X}_v; \theta_G)$. We train D to maximize the probability of assigning the correct label to both neighborhood nodes representations and

embeddings from G. We simultaneously train G to minimize $\log(1 - D(G(\mathcal{X}_v; \theta_G))$.

LSTM: The node representation dimension parameter d in NEGAN is set according to requirements. The input of the discriminator should be same dimension. Therefore, this paper encodes neighborhood nodes feature to d-dimensional representation, since the output of the generator is also a d-dimensional representation. As we know, Long Short Term Memory networks, usually just called LSTM, is a special kind of RNN remembering information for long periods of time, thus we combine GAN with LSTMs. In our work, we use node2vec sampling strategy to get neighborhood nodes N_S, then generate neighborhood nodes representaion $f(\mathcal{X}_N; \theta_f)$ in d-dimensional space with LSTM.

We train a GAN as a deep neural network that parametrically learns the node representation based on the node content feature. Given the pair (v, N_S), the input feature for both branches are the corresponding content feature $\mathcal{X}_v$ of target node v and $\mathcal{X}_N$ of neighborhood nodes N_S. Here we use the multi-layer perceptron (MLP) in the generator and discriminator where the h-th hidden layer is given by

$$z_h(\mathcal{X}_i) = g_h(W_h \cdot z_{h-1}(\mathcal{X}_i) + b_h), \quad 1 \leq h \leq H \qquad (3)$$

where $g(\cdot)$ is the element-wise activation function. In the first $H - 1$ layers, the activation functions are all *sigmoid* function [28].

For the H-th hidden layer, its output is used as the final node representation. Thus, linear activation function is used, where $z_H(\mathcal{X}_i) = W_H \cdot z_{H-1}(\mathcal{X}_i) + b_H$. We refer one MLP as the generator, denoted by $G = z_H(\cdot)$, and the node representation for node v will be $G(\mathcal{X}_v; \theta_G)$. At the same time, We refer another MLP as the discriminator, denoted by $D = z_H(\cdot)$, and the judgment value will be $D(G(\mathcal{X}_v; \theta_G); \theta_D)$ and $D(f(\mathcal{X}_N; \theta_f); \theta_D)$.

Thereafter, samples synthesized by the generator are evaluated by the discriminator. The generator is typically a multi-layer perceptron neural network, and the discriminator is also a multi-layer perceptron neural network.

We define the energy function on the top of the model, between $D(G(\mathcal{X}_v; \theta_G); \theta_D)$ and $D(f(\mathcal{X}_N; \theta_f); \theta_D)$.

$$\min_G \max_D = \mathbb{E}[\log D(f(\mathcal{X}_N; \theta_f); \theta_D)] + \mathbb{E}[\log D(1 - G(\mathcal{X}_v; \theta_G); \theta_D)], \quad v \in \mathcal{V} \qquad (4)$$

The purpose of GAN is to make the node and neighborhood nodes as close as possible in the representation space. We train discriminator and generator simultaneously, then we train the generator separately in order to generate a similar representation of the neighborhood nodes. We will show that Algorithm 2 optimizes Eq.(4), thus we obtain the desired result.

C. Training

We adopt the adaptive gradient algorithm (AdaGrad) [29] to train our generative adversarial networks architecture in the mini-batch mode. We first sample a batch of pairs $(\mathcal{X}_v, \mathcal{X}_N)$ and take a gradient step to optimize the loss function Eq.(4).

In order to solve the problem of trivial values fed forward to the output layer caused by sparse feature matrix $\mathcal{X}$, batch normalization [30] is applied to each hidden layer of both generator and discriminator except the last output layer. The batch normalization helps stabilize the learning process by normalizing the input to each unit to have zero mean and unit variance. Additionally, several typical activation functions have been evaluated, such as sigmoid, tanh and relu. Eventually, sigmoid has a consistent better performance comparing with other activation functions.

IV. EVALUATION

In this section, we evaluate the node representation learned by the proposed method NEGAN on two publicly accessible datasets. Specifically, we choose three application scenarios for experiments, i.e., node classification, community detection, and representation visualization. NEGAN is compared with several state-of-the-art network embedding methods.

A. Datasets

The purpose of method NEGAN is learning from both the network structures and the node content feature. We select citation network and information network as our experimental datasets. They are Pubmed and Wiki used in [31].

1) ***Pubmed****:* Pubmed is a research paper citation network. There are 19,717 scientific papers from three distinct research areas. The papers are treated as nodes in the network, and the research areas are treated as class labels. There are 44,338 edges among nodes. The content vector is short texts generated from the papers content, and they are represented by a binary-valued word vector. The dimension of the content vector is 500.

2) ***Wiki****:* Wiki dataset is different from Pubmed. It contains 2,405 web pages categorized into seventeen classes. There are 17,981 hyperlinks among these web pages. Wiki dataset stores the webpage content by using the term frequency-inverse document frequency (TF-IDF) vector. The TF-IDF vector is 4,973 dimension, indicating 4,973 unique words.

The details of the networks are summarized in Table I. We respectively construct an undirected graph based on the edges in each of mentioned datasets. The TF-IDF vectors or binary-valued vectors form the node content feature $\mathcal{X}$.

TABLE I: Dataset statistics

Dataset	Pubmed	Wiki
Content feature	Binary	TF-IDF
Type	Citation	Hyperlink
Nodes	19,717	2,405
Edges	44,338	17,981
Classes	3	17
Feature	500	4,973

Algorithm 2 Minibatch adaptive gradient training of NEGAN. The number of training discriminator and generator is a hyperparameter

for number of training discriminator and generator iterations **do**
  Sample minibatch of m node feature samples $\{\mathcal{X}^{(1)}, ..., \mathcal{X}^{(m)}\}$ and corresponding neighborhood node feature samples $\{\mathcal{X}_N^{(1)}, ..., \mathcal{X}_N^{(m)}\}$
  Update the discriminator by ascending its stochastic gradient: $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \log \left[D(\mathcal{X}_N^{(i)}) + \log(1 - D(G(\mathcal{X}^{(i)}))) \right]$
  Update the discriminator by ascending its stochastic gradient: $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left[D(G(\mathcal{X}^{(i)})) \right]$
end for
for number of training generator iterations **do**
  Sample minibatch of m node feature samples $\{\mathcal{X}^{(1)}, ..., \mathcal{X}^{(m)}\}$
  Update the discriminator by ascending its stochastic gradient: $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log \left[D(G(\mathcal{X}^{(i)})) \right]$
end for

TABLE II: *Averaged Micro-F1, Macro-F1 scores for node classification task on Wiki Network.*

Dataset		Wiki					
Method		Node Feature	DeepWalk	node2vec	Naive Comb	TADW	NEGAN
micro	10%	60.23	57.64	58.24	63.16	66.48	**70.29***
	30%	69.18	65.48	66.03	71.02	73.56	**75.71***
	50%	72.65	68.61	70.98	74.12	75.36	**76.80***
	70%	74.93	69.02	71.05	76.42	77.47	**81.19***
macro	10%	38.76	40.55	44.14	47.08	48.56	**53.03***
	30%	54.21	50.26	51.02	54.55	62.54	**66.46***
	50%	60.24	55.20	58.14	66.61	70.24	**72.85***
	70%	65.67	59.03	58.49	68.93	71.51	**75.63***

TABLE III: *Averaged Micro-F1, Macro-F1 scores for node classification task on Pubmed Network.*

Dataset		Pubmed					
Method		Node Feature	DeepWalk	node2vec	Naive Comb	TADW	NEGAN
micro	10%	80.97	75.42	78.41	81.61	81.53	**82.91***
	30%	83.98	77.23	80.03	84.48	84.29	**85.71***
	50%	85.29	78.62	80.92	84.16	84.76	**85.80***
	70%	85.42	79.18	81.76	83.71	83.82	**85.92***
macro	10%	80.78	73.71	78.41	80.83	81.11	**84.03***
	30%	84.07	75.77	78.58	83.18	83.29	**85.49***
	50%	85.29	77.24	79.77	84.71	84.47	**85.87***
	70%	85.42	79.18	81.35	84.29	84.54	**85.99***

B. Baseline Methods

We compare NEGAN with several state-of-the-art network embedding methods, each method will take the implementation released by the original authors:

- **Node Feature:** Node content feature is directly treated as node representation. To fairly compare with other methods, the typical dimension reduction method SVD is used for dimension reduction.
- **DeepWalk:** DeepWalk [9] is the first network representation learning method inspired from neural language models. For each node, the random walk is used to obtain the network contextual information. Then, it uses the structual context for learning network embedding.
- **node2vec:** node2vec [10] is a variant of DeepWalk [9]. It designs a second-order random walk, which provides a trade-off between breadth-first (BFS) and depth-first (DFS) graph searches, and hence it produces higher-quality and more informative embeddings than DeepWalk.
- **Naive Combination:** The text feature vector of a node and node2vec [10] embedding of this node are concatenated as its new representation. This is the most straightforward way to integrate graph topology and node content feature. The representation dimension is 512 (256 from node2vec embedding and 256 from reduced node content feature).
- **TADW:** TADW [12] factorizes a matrix to simulate DeepWalk. It utilizes both graph structure information and node content feature to build graph representation.

Note that Node Feature only uses the node content information, DeepWalk and node2vec are network-only representation learning methods, while Naive Combination and TADW utilize both network structure and content feature. The proposed NEGAN also utilizes two types of information, but it integrates them based on the generative adversial nets. The node representation dimension parameter d in NEGAN is set to 256 if there is no specification, which is same as that in DeepWalk, node2vec and TADW for conducting fair comparisons. The representation dimension of Navie Combination is set to 512 (256 from node2vec embedding and 256 from reduced node content feature) if there is no specification.

C. Experiment Setups

For all datasets, we use a three-layer MLP (H =3) in generator, the output layer contains 256 units, thus produces node representation in 256-dimension. What's more, we also use a three-layer MLP (H=3) in discriminator to distinguish the representation from generator and the representaion from LSTM. The part LSTM contains four LSTM cells, which encodes neighborhood node feature in 256-dimension.

D. Node Classification

Table II and Table III show the micro-F1 and macro-F1 scores obtained by each method on the classification task. Upon getting the node representations, we train the one-vs-rest Support Vector Machine classifier implemented by scikit-learn for all models [32]. We repeat the experiments for ten times, and report average performance for each method. In all evaluation experiments, we use different ratios of labeled data and compare the performance of all evaluated approaches at different settings. First and foremost, NEGAN outperforms all baselines on both datasets. This demonstrates that NEGAN can still effectively encode the information of vertices into the learned representations.

The performance of NEGAN is followed by that of TADW and Naive combination which integrate network topology with node content feature. DeepWalk, node2vec and Node feature which use only the network structure or node content information perform worst. This further verifies the positive effects of integrating the graph topology and node content feature into the network representation learning. Moreover, modeling them can lead to better network embedding process and benefit downstream applications.

TABLE IV: *Averaged NORMALIZED MUTUAL INFORMATION scores for community task on Wiki Network.*

Dataset	Wiki	Pubmed
Node Feature	0.3002	0.3134
DeepWalk	0.3701	0.2606
node2vec	0.3751	0.2665
Naive Comb	0.3272	0.2673
TADW	0.3953	0.2214
NEGAN	$\mathbf{0.4210}^*$	$\mathbf{0.3346}^*$

E. Community Detection

The aim of community detection task is to predict the most possible community assignment to each node. We use k-means algorithm [33] to cluster nodes for community detection, and the number of clusters K in k-means algorithm is set the same as the number of distinct labels in Wiki dataset. Once node ground truth labels are known, the correspondence between the detected and ground-truth communities can be measured by *Normalized Mutual Information* (NMI). NMI [34] is used to measure the similarity between detected clusters and ground truth clusters. The score ranges between 0 and 1. A higher score indicates that detected clusters are more consistent to ground truth.

The community detection results are presented in Table IV. As we see from the table, the performance of NEGAN is the best in both metrics. The community detection task is difficult in Wiki dataset, since it has 17 labeled communities. When using NMI, NEGAN outperforms TADW for over 0.03. In Pubmed dataset, all baseline methods perform similarly to each other, while NEGAN outperforms all other methods with a margin over 0.02. The community detection results indicate that the attributed embeddings learned by NEGAN are more effective than all other methods, even when processing a dataset with a great quantity of communities.

F. Visualization and Clustering

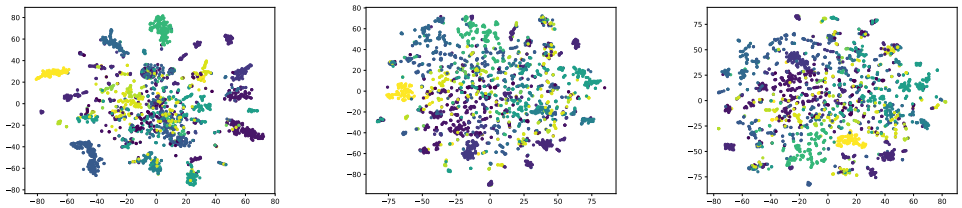
To the best of our knowledge, the meaningful visualization generated from network embedding is strongly useful, since it can map a network in a low dimensional space (usually 2-dimension). Moreover, visualization is a way to inspect the effect of the embeddings clustering. The results are shown in Figure 2, where nodes are colored by their ground truth labels. As we can see in Figure 2b and Figure 2c, using only network topology information doesn't perform well. Most of the nodes with different class labels are overlapped. This implies that the nodes nearby are correlated to the label, but nodes faraway can also have the same labels. The result of TADW in Figure 2e is better than that of node feature, DeepWalk and node2vec, but it still has nodes with different labels overlapped in the center area. This is because TADW suffers from the problem raised by node feature. The results of the Naive combination and NEGAN are more separable in visualization, but the dimension of Naive combination embedding is twice as long as the dimension of NEGAN embedding. However, the results of NEGAN are more converged and separable because of the contrastive information learned in the generative adversial networks. As a result, NEGAN performs better and generates a more meaningful layout of the network than other methods.

G. Parameter Sensitivity

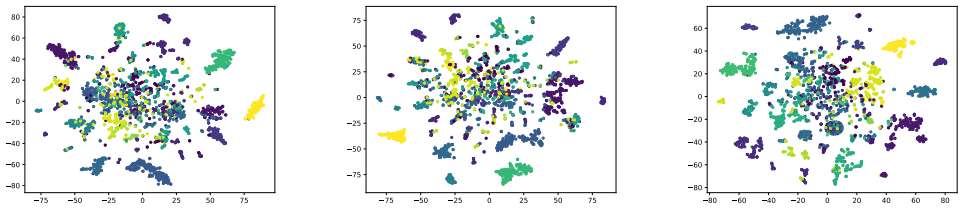
NEGAN has two important hyperparameters: the node representation dimension d, the number of layers in the GAN network H .

We fix training ratio to 70 and test classification accuracies with different d (representation dimension) on the Wiki dataset. We compare the Micro-F1 and Macro-F1 measure of NEGAN with all other baseline methods. The performance of different methods for the setting of d is shown in Figure 3. As well as the performance of DeepWalk and node2vec, NEGAN gets its peak at $d = 256$, while TADW gets its peak when d is 1024. What'more, the result is the best when its dimension is reduced to 256 in the method of Node feature. As we can see from the figure, NEGAN consistently has the best performance in different representation dimensionality settings.

We also evaluate the the number of layers H in GAN on two datasets for node classification. The results from Figure 4 show the performance of NEGAN is the best when the number of layers is 3. However, when the number of layers is over 3, the Micro-F1 and Macro-F1 begin to drop. This is because deeper



(a) Node Feature (b) DeepWalk (c) node2vec



(d) Naive Combination (e) TADW (f) NEGAN

Fig. 2: Visualization of 8000 randomly selected nodes in Wiki Dataset. (a), (b), (c), (d), (e) and (f) are the visualization result for node feature, DeepWalk, Node2vec, Naive Combination, TADW and NEGAN respectively.

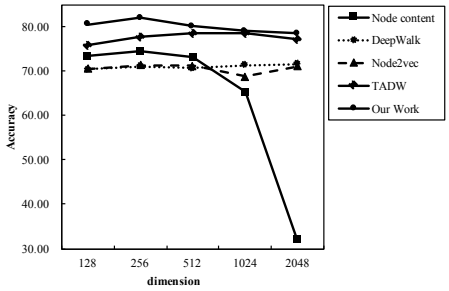


Fig. 3: The sensitivity of representation dimension d for different methods in Micro-F1 metric.

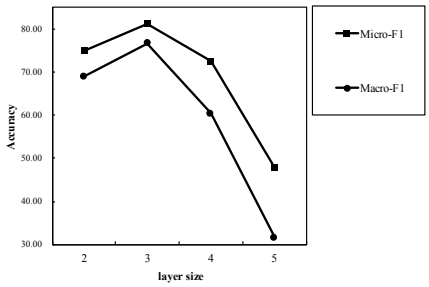


Fig. 4: Parameter sensitivity for the number of layers H.

networks can learn more information, but deeper networks will also be more costly for learning.

V. Conclusion

To learn informative embeddings for real-world networks, it is crucial to account for both network structure and content information. For this purpose, We propose a novel parametric representation learning method, NEGAN, which maps graph nodes into a low-dimensional space by integrating the network topology and node content information. Extensive experiments show that NEGAN can learn informative representations for real-world networks and achieve superior performance on the tasks of node classification, community detection and visualization, comparing with state-of-the-art methods like DeepWalk, node2vec and TADW.

In our future work, we will be devoted to exploring how to optimize the proposed NEGAN deep neural networks to learn more comprehensive representations from multi-source enriched networks, considering not only node content feature, network topology, but also edge content.

VI. Acknowledgement

The authors would like to thank the anonymous reviewers for their helpful comments. This work was supported by National Key R&D Program of China (2017YFC0803700), National Natural Science Foundation of China (61502320) and Science Foundation of Shenzhen City in China.

References

[1] P. Wang, G. Robins, P. Pattison, and E. Lazega, "Social selection models for multilevel networks," *Social Networks*, vol. 44, pp. 346–362, 2016. [Online]. Available: https://doi.org/10.1016/j.socnet.2014.12.003

[2] V. Karwa and S. Petrovic, "Coauthorship and citation networks for statisticians: Comment," *CoRR*, vol. abs/1608.06667, 2016. [Online]. Available: http://arxiv.org/abs/1608.06667

[3] M. Kanehisa, S. Goto, M. Furumichi, M. Tanabe, and M. Hirakawa, "KEGG for representation and analysis of molecular networks involving diseases and drugs," *Nucleic Acids Research*, vol. 38, no. Database-Issue, pp. 355–360, 2010. [Online]. Available: https://doi.org/10.1093/nar/gkp896

[4] Q. Dai, Q. Li, J. Tang, and D. Wang, "Adversarial network embedding," *CoRR*, vol. abs/1711.07838, 2017. [Online]. Available: http://arxiv.org/abs/1711.07838

[5] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "Graphgan: Graph representation learning with generative adversarial nets," *CoRR*, vol. abs/1711.08267, 2017. [Online]. Available: http://arxiv.org/abs/1711.08267

[6] L. Liao, X. He, H. Zhang, and T. Chua, "Attributed social network embedding," *CoRR*, vol. abs/1705.04969, 2017. [Online]. Available: http://arxiv.org/abs/1705.04969

[7] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *CoRR*, vol. abs/1705.02801, 2017. [Online]. Available: http://arxiv.org/abs/1705.02801

[8] 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, 2017. [Online]. Available: http://sites.computer.org/debull/A17sept/p52.pdf

[9] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, 2014, pp. 701–710. [Online]. Available: http://doi.acm.org/10.1145/2623330.2623732

[10] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 2016, pp. 855–864. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939754

[11] C. Tu, W. Zhang, Z. Liu, and M. Sun, "Max-margin deepwalk: Discriminative learning of network representation," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 2016, pp. 3889–3895. [Online]. Available: http://www.ijcai.org/Abstract/16/547

[12] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 2015, pp. 2111–2117. [Online]. Available: http://ijcai.org/Abstract/15/299

[13] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Homophily, structure, and content augmented network representation learning," in *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*, 2016, pp. 609–618. [Online]. Available: https://doi.org/10.1109/ICDM.2016.0072

[14] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, 2017, pp. 203–209. [Online]. Available: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14589

[15] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial networks," *CoRR*, vol. abs/1406.2661, 2014. [Online]. Available: http://arxiv.org/abs/1406.2661

[16] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *CoRR*, vol. abs/1310.4546, 2013. [Online]. Available: http://arxiv.org/abs/1310.4546

[17] 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, WWW 2015, Florence, Italy, May 18-22, 2015*, 2015, pp. 1067–1077. [Online]. Available: http://doi.acm.org/10.1145/2736277.2741093

[18] K. Tu, P. Cui, X. Wang, F. Wang, and W. Zhu, "Structural deep embedding for hyper-networks," *CoRR*, vol. abs/1711.10146, 2017. [Online]. Available: http://arxiv.org/abs/1711.10146

[19] 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, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 2016, pp. 1895–1901. [Online]. Available: http://www.ijcai.org/Abstract/16/271

[20] L. van der Maaten, "Accelerating t-sne using tree-based algorithms," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3221–3245, 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2697068

[21] Y. Bengio, A. C. Courville, and P. Vincent, "Unsupervised feature learning and deep learning: A review and new perspectives," *CoRR*, vol. abs/1206.5538, 2012. [Online]. Available: http://arxiv.org/abs/1206.5538

[22] S. Chang, W. Han, J. Tang, G. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, 2015, pp. 119–128. [Online]. Available: http://doi.acm.org/10.1145/2783258.2783296

[23] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, 2016, pp. 1145–1152. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12423

[24] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang, "IRGAN: A minimax game for unifying generative and discriminative information retrieval models," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, 2017, pp. 515–524. [Online]. Available: http://doi.acm.org/10.1145/3077136.3080786

[25] Y. Zhang, R. Barzilay, and T. S. Jaakkola, "Aspect-augmented adversarial networks for domain adaptation," *TACL*, vol. 5, pp. 515–528, 2017. [Online]. Available: https://transacl.org/ojs/index.php/tacl/article/view/1116

[26] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, 2017, pp. 2852–2858. [Online]. Available: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14344

[27] J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, and D. Jurafsky, "Adversarial learning for neural dialogue generation," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, 2017, pp. 2157–2169. [Online]. Available: https://aclanthology.info/papers/D17-1230/d17-1230

[28] Y. LeCun, L. Bottou, G. B. Orr, and K. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade - Second Edition*, 2012, pp. 9–48. [Online]. Available: https://doi.org/10.1007/978-3-642-35289-8_3

[29] J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=2021068

[30] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, 2015, pp. 448–456. [Online]. Available: http://jmlr.org/proceedings/papers/v37/ioffe15.html

[31] Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, 2016, pp. 40–48. [Online]. Available: http://jmlr.org/proceedings/papers/v48/yanga16.html

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. VanderPlas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *CoRR*, vol. abs/1201.0490, 2012. [Online]. Available: http://arxiv.org/abs/1201.0490

[33] S. Fortunato, "Community detection in graphs," *CoRR*, vol. abs/0906.0612, 2009. [Online]. Available: http://arxiv.org/abs/0906.0612

[34] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *IEEE Trans. Neural Networks*, vol. 20, no. 2, pp. 189–201, 2009. [Online]. Available: https://doi.org/10.1109/TNN.2008.2005601