Network Structure and Transfer Behaviors Embedding via Deep Prediction Model

Xin Sun,[1] Zenghui Song,[1] Junyu Dong,[1] Yongbo Yu,[1] Claudia Plant,[2,3] Christian Böhm[4]
[1]Department of Computer Science and Technology, Ocean University of China, Qingdao, China
[2]Faculty of Computer Science, University of Vienna, Vienna, Austria
[3]Data Science @ University of Vienna, Vienna, Austria
[4]Ludwig-Maximilians-Universität München, Munich, Germany
{sunxin, dongjunyu}@ouc.edu.cn, {szh, yuyongbo}@stu.ouc.edu.cn, claudia.plant@univie.ac.at, boehm@ifi.lmu.de

Abstract

Network-structured data is becoming increasingly popular in many applications. However, these data present great challenges to feature engineering due to its high non-linearity and sparsity. The issue on how to transfer the link-connected nodes of the huge network into feature representations is critical. As basic properties of the real-world networks, the local and global structure can be reflected by dynamical transfer behaviors from node to node. In this work, we propose a deep embedding framework to preserve the transfer possibilities among the network nodes. We first suggest a degree-weight biased random walk model to capture the transfer behaviors of the network. Then a deep embedding framework is introduced to preserve the transfer possibilities among the nodes. A network structure embedding layer is added into the conventional Long Short-Term Memory Network to utilize its sequence prediction ability. To keep the local network neighborhood, we further perform a Laplacian supervised space optimization on the embedding feature representations. Experimental studies are conducted on various real-world datasets including social networks and citation networks. The results show that the learned representations can be effectively used as features in a variety of tasks, such as clustering, visualization and classification, and achieve promising performance compared with state-of-the-art models.

Introduction

Nowadays structured data in the form of networks is ubiquitous in our daily lives and has an astonishing growth. Especially the pervasive use of online social networks, such as Facebook and Twitter, generates huge network data continuously at unprecedented rates. New applications, such as node classification (Sen et al. 2008), link prediction (Liben-Nowell and Kleinberg 2007), and social role discovering (Henderson et al. 2012), arise in various areas. For example, node classification and link prediction are generally used for similar user searching and advertisement recommendation. However, the vast majority of existing machine learning algorithms for classification and prediction are feature-based, which means informative and discriminating attribute-value entities are required. Rather than developing special learning algorithms for network data, it is more practical to learn feature vector representations for nodes and edges of the network. Once the vectorized representation is obtained, the data mining tasks for large networks can be well solved by state-of-the-art machine learning algorithms. Therefore, the issue of how to transfer the link-connected nodes of the huge network into feature representations is critical.

Intuitively the widely studied graph embedding approaches including IsoMap (Tenenbaum, De Silva, and Langford 2000), LLE (Roweis and Saul 2000), and Laplacian Eigenmaps (Belkin and Niyogi 2001) seem to be good solutions for the network feature learning problem. Nevertheless, the graph representation, which is the core of these approaches, is directly derived from the data itself and perfectly reflects the global and local structure of the data. On the contrary, natural networks encountered in the real world are sparse and have many undiscovered and noisy links (Tang and Liu 2012). We cannot get a full view of the authentic relationship between any two nodes. So it is unpractical to get features from the network itself by traditional graph embedding approaches. One typical solution is to form hand-engineering features for each node by observing its interaction behavior with neighbors (Perozzi, Al-Rfou, and Skiena 2014), such as first-order proximity and second-order proximity (Tang et al. 2015). However, the neighbor context information is still not enough to well describe the node due to the undiscovered and noisy links. The networks from the real world exhibit significant dynamic evolution behavior. It is impossible to catch the network evolution process from the structured data itself. Fortunately, the dynamic evolution process can be reflected by the transfer behaviors among the nodes, such as rumor propagation and reputation transmission.

Previous work did not systematically consider local and global structure with its dynamical behaviors, so this paper proposes a Deep Network Embedding framework to solve this issue in two aspects. One is how to capture the network structure and neighborhood context information from the network structured data. The other is how to embed the complex and non-linear network structural data into low-dimensional vector representations, while preserving the transfer possibilities among the nodes.

Our contributions are as follows:

- We propose a degree-weight biased random walk model to preserve the global network context information. It captures the node's roles and the information transfer behaviors among nodes.

- To preserve the transfer possibilities among the nodes, we design a deep network embedding framework, which suggests a network structure embedding layer into the conventional Long Short-Term Memory Network (LSTM) to utilize its sequence prediction ability.
- We leverage a Laplacian supervised embedding space optimization to capture the local network structure, which makes the connected nodes close with each other in the low-dimensional space.
- We conduct extensive experiments on various tasks with real-world datasets, and achieve competitive performance.

Related Work

This work focuses on engineering features from the network structured data. A practical way to engineer features for the network structured data is to design domain-specific features based on expert knowledge. For example, Henderson et al. (Henderson et al. 2011) introduced the statistical properties of the node itself (e.g., degree) and its neighborhood (number of edges) to generate the recursive structural features. With their structural features, a role discovery approach RolX (Henderson et al. 2012) is designed to mine the similar nodes with similar social behavior. Inspired from the representational learning for natural language processing, such as Skip-gram model (Mikolov et al. 2013), Perozzi et al. (Perozzi, Al-Rfou, and Skiena 2014) proposed the DeepWalk method by treating nodes in the network as words. Their motivation is that the distributions of vertices in social network and words in natural language appearing in short random walks both follow a power-law behavior. Similarly, node2vec (Grover and Leskovec 2016) offers flexibility random walk sampling strategies to capture the diversity of the neighborhood. However, these methods do not make clear what kind of network properties are preserved. LINE (Tang et al. 2015) method defines the first-order and second-order proximities to clearly preserve both the local and global network structures. HOPE (Ou et al. 2016) solves the asymmetric transitivity problem in a directed network and preserves high-order proximities of large-scale graphs with generalized SVD.

The network data is highly non-linear (Tang and Liu 2012) with various links, Tian et al. (Tian et al. 2014) proposed a deep autoencoder to learn nodes' representation which was used for graph clustering. And Wang et al. (Wang, Cui, and Zhu 2016) designed a semi-supervised deep model to capture the highly non-linear network structure based on the first-order and second-order proximity. DNGR (Cao, Lu, and Xu 2016) use the stacked denoising autoencoder to learn the low-dimensional node representations.

As the convolutional neural network (CNN) has achieved great success in the image processing area, some works were carried out with CNN (Bruna et al. 2013). Inspired by spectral convolutions, Kipf et al. (Kipf and Welling 2016) proposed an efficient convolutional neural network that can classify the network data. Furthermore, Niepert et al. (Niepert, Ahmed, and Kutzkov 2016) tried to construct a convolution operator from spatial domain instead of spectral domain for the network data.

Besides the CNN deep model, Generative Adversarial Networks (GAN) (Goodfellow et al. 2014) was also introduced to handle the network structured data. For instance, Wang et al. (Wang et al. 2017) and Dai et al (Dai et al. 2017) both use the GAN model to learn the network representation. Although the above methods have achieved good performance on some tasks, the structure of the network they considered are not comprehensive. This section can not be a complete review of all algorithms and just briefly describes the most related studies.

Problem Definition

In this section, we define the feature engineering problem for network data and give some concepts of network structured data concerned in this paper.

Definition 1. The network structured data can be represented as N=$\{V,\, E\}$. V=$\{v_1,\, v_2,\, \cdots,\, v_n\}$ is the node set where n is the number of nodes. $E = \{e_{ij}\}_{i,j=1}^{n}$ is the edge set where e_{ij} denotes the edges from node v_i to v_j. A weight $w_{ij} \geq 0$ is attached to each edge e_{ij}, and $w_{ij} = 0$ if v_i and v_j are not directly connected. For unweighted network $w_{ij} = 1$, and for undirected network $w_{ij} = w_{ji}$.

Definition 2. (*Network numerical formalization*) The network feature representation learning problem aims to find a robust and low-dimensional vector presentation for each node. This paper addresses this problem in two steps: (1) represent the structural network data in the numerical vector space that capture the structural function and global neighborhood context information; (2) encode the complex and non-linear network structural representations into low-dimensional vector representations, while preserving the transfer possibilities among the nodes. Then this problem can be defined as follows.

Given a network N=$\{V,\, E\}$, numerical formalization aims to represent each vertex $v_i \in V$ as D-dimensional numerical vector $x_i \in \mathbb{R}^D$. The numerical vector x_i preserves the neighborhood and structural role information. (*Feature learning*) It aims to learn a function f: $x_i \rightarrow y_i \in \mathbb{R}^d$, where $d \ll D$. The objective of the function f is to minimize the distance between y_i and y_j if they are similar to each other.

In the next section, we will introduce the network numerical formalization and feature learning respectively.

Model Description

In this section, we introduce a Deep Network Embedding framework (DNE) for representation learning as shown in Figure 1. The DNE framework consists of three parts: (i) Degree-Weight biased random walk (DW-$RandomWalk$) for sampling sequences, (ii) A novel network structure embedding layer of the Long Short-Term Memory (LSTM) deep model (Graves 2013) for encoding high-dimensional space into low-dimensional one, (iii) Laplacian (Belkin and Niyogi 2001) supervised embedding space optimization (LapEO) for capturing the local network structure.

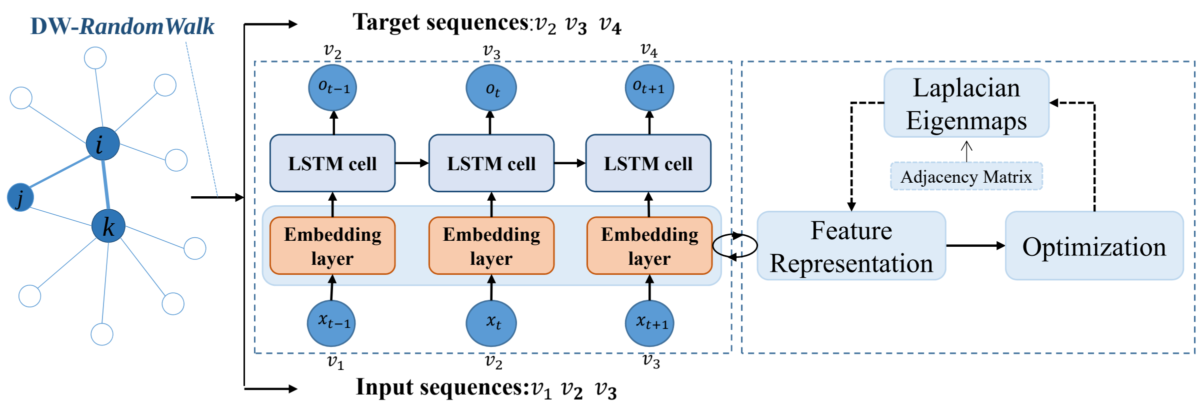


Figure 1: The proposed Deep Network Embedding (DNE) framework

Degree-Weight biased Random Walk

In most of the real networks, edge usually implies the similarity of two nodes, such as friend relationships. Some literatures (Wang, Cui, and Zhu 2016) define the first-order proximity to characterize the pairwise proximity. The drawback is that the social role and relationship information are missing. As shown in Figure 2, the social role of nodes i and k is much more similar than i and j in term of function and structure. The information transfer more possibly exists between nodes i and k. Here we propose a degree-weight biased proximity to characterize the transfer priority.For each pair of nodes connected by an edge e_{ij}, the proximity s_{ij} is calculated as follow.

$$s_{ij} = w_{ij} \cdot \frac{\min(d_i, d_j)}{\max(d_i, d_j) + \alpha} \tag{1}$$

The degree-weight biased proximity has the ability to capture the social role information among nodes. For example, we can see that proximity s_{ik} and s_{ij} in Figure 2 is characterized correctly. That means nodes with similar function and role have high proximity probability. The α is an adjustment parameter which makes the s_{ij} smooth. We use degree-weight biased proximity s_{ij} to guide the walking process. It tends to capture the primary structure characteristics among the node. The transition probability is defined as follows:

$$P(u^i \to u^{i+1}) = s_{u^i, u^{i+1}}, u^{i+1} \in \mathcal{N}(u^i) \tag{2}$$

where u^i, $u^(i+1)$ denote two connected nodes in networks. And the transition probabilities of each node are normalized in order to make the summation equaling to 1. We call this sampling method Degree-Weight biased Random Walk. Nowadays, there are many excellent sampling methods on network representation learning task, e.g., truncated random walk (Perozzi, Al-Rfou, and Skiena 2014) and biased random walk (Grover and Leskovec 2016). We will demonstrate that our method is more suitable in the later section.

The Embedding Model

The sequences generated by random walk can be regarded as time series, i.e., there is a temporal correlation among the nodes in the generated sequence. In other words, such sequences reflect the transfer behaviors among the network nodes and can be predicted. Inspired by this, we propose to employ a prediction model, i.e., LSTM, to process such sequences. LSTM is a kind of model well-suited to learn from experience to process and predict time series. It was originally proposed by Sepp Hochreiter and Jürgen Schmidhuber (Hochreiter and Schmidhuber 1997) in 1997, and is suitable for the speech recognition problem with huge vocabulary of words.

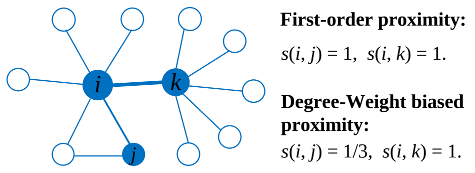

First-order proximity:
s(i, j) = 1, s(i, k) = 1.
Degree-Weight biased proximity:
s(i, j) = 1/3, s(i, k) = 1.
i
k
j

Figure 2: A toy example of subgraphs. Social role and relationship is quite different among nodes i, j and k.

The conventional LSTM model is popularly applied to natural language processing, mainly for sequence prediction, e.g., predicting the next word or sentence (Sutskever, Martens, and Hinton 2016) and translating English sentence to another language (Sutskever, Vinyals, and Le 2014). In such areas, researchers pay attention to the final result of these tasks. However the network representation learning is concerned with representation of hidden layer (Tian et al. 2014).

For the aim of network representation learning, we suggest a new network structure embedding layer into the conventional LSTM Network to utilize its sequence prediction ability. The LSTM model itself is trained to predict the next step in the sequences. And the embedding layer is used to learn the representation, which maps high-dimensional data to low-dimensional space. Let $g : \mathcal{C} \longmapsto \mathcal{X}$, $\mathcal{X} \in \mathbb{R}^{|V| \times d}$, $d \ll |V|$, denotes the embedding layer where $\mathcal{C}$ is the corpus we built by random walk, $\mathcal{X}$ is the d-dimensional vector we expected.

For the LSTM traning, we use the sequence generated by DW-$RandomWalk$ to predict its next node. It is worth noting that this is a $fake$ task (Mikolov et al. 2013). We just want to obtain the nodes' representation through predictive procedure. The embedding layer is the real output we expect.

For network data, our goal is to maximize the likelihood of predicting the next node, i.e,:

$$\mathcal{P}(X^{t+1}|X^t) \tag{3}$$

where $X^t = (x^1, x^2, \dots, x^t), X^{t+1} = (x^2, x^3, \dots, x^{t+1})$. We add a softmax function as $softmax(x_i) = \frac{e^{x_i}}{\sum_i e^{x_i}}$ after the output layer. For each input sequence, the goal is to predict its next node, so we use cross entropy to evaluate the loss:

$$\mathcal{L} = -\sum \mathcal{Y} \ln \mathcal{O} \tag{4}$$

where $\mathcal{Y}$ is the predicted target node, and $\mathcal{O}$ is the output of the softmax layer.

We use Adaptive Moment Estimation (Adam) (Kingma and Ba 2014) to optimize the objective function and Backpropagation (Williams and Peng 1990) to update parameters including the representation Φ.

Embedding Space Optimization

The embedding model is able to capture the global structural information and transition context of the network. Because it not only considers the information at time t_c, but also considers the information before t, e.g. $t_{c-1}, t_{c-2,\dots}$. However, it does not preserve the local structure of the network. We should guarantee the connected nodes still close with each other in the new embedding space. To this end, we propose a Laplacian supervised Embedding space Optimization (LapEO) to preserve the local network structure. The idea is motivated from Laplacian Eigenmaps (Belkin and Niyogi 2001), which attempts to make connected nodes as close as possible in low-dimensional space. Network representation learning is to learn a function to map the data in high-dimensional space to low-dimensional space and maintains structural consistency (Grover and Leskovec 2016). In other words, the neighborhood context of each node in the original space and the new space should be as similar as possible. So we propose the loss function of the Laplacian optimization as follow.

$$\mathcal{L}_{reg} = \sum_{i,j} (y_i - y_j)^2 A_{ij} = 2 \cdot Tr(Y^T L Y) \tag{5}$$

where $Y \in \mathbb{R}^{|V| \times d}$ is the node representation, A is the adjacency matrix, $L = D - A$ is Laplacian matrix, $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix, $D_{i,i} = \sum_j A_{i,j}$.

We optimize the loss functions $\mathcal{L}$ and $\mathcal{L}_{reg}$ in an Expectation-Maximization-like iterative way. That means we choose an individual way and use different optimization algorithms. There are two main reasons. The first one is that the parameters of LSTM model are difficult to update when there are two loss functions. The other reason is that the representation is shared between these two stages. The embedding space optimization can be performed after each LSTM epoch to accelerate the LSTM training procedure.

Experiments

In this section, we empirically validate the effectiveness of the proposed algorithm in comparison to various state-of-the-art algorithms.

Datasets

The following datasets are employed for our experiments. The datasets contain various numbers of nodes and edges which could give sufficient validation for all the methods, especially the Pubmed and BlogCatalog are large networks.

- BlogCatalog (Tang and Liu 2009) is social network dataset about blogger authors. It commonly be used as the ground-truth for multi-label classification task.
- CoRA, CiteSeer and PubMed (Sen et al. 2008) are collections of scientific publications from different databases.
- The 20-Newsgroups (Lang 1995) dataset is a collection of 20,000 newsgroup documents, partitioned into 20 different categories.

Baseline Methods

We compare our method with several baseline methods, including DeepWalk (Perozzi, Al-Rfou, and Skiena 2014), LINE (Tang et al. 2015), SDNE (Wang, Cui, and Zhu 2016), Struc2Vec (Ribeiro, Saverese, and Figueiredo 2017), and GraphGAN (Wang et al. 2017). There are also some other embedding methods, however, we can not show all of them here. The above methods are recently proposed and give extensive experiments with other methods in the corresponding papers.

Parameter Settings

For the compared methods, we set the optimal parameters as suggested in their original papers. For example, DeepWalk, we set window size as 10, walk length as 40 and walks per vertex as 40. For LINE, the number of negative samples is set as 5 and the total number of samples is 10 billion. For SDNE, we set the number of layers in the model as 3, the hyper-parameter α and β as 0.1 and 10. For Struc2Vec, we set window size as 10, walk length as 80, walks per vertex as 10. All methods get representation with dimensions of 128. For our method, we set walk length to be 100. We use different γ(walks per node) for different datasets. For BlogCatalog and Pubmed datasets, we set walks per node as 30. For other datasets, we set walks per node as 100. LSTM learning rate is 0.001. For convenience, we set LSTM timesteps equal to walk length l .

Experiments Results

In this section, we evaluate the learned representations of different methods through three downstream tasks: clustering, classification and visualization.

Clustering Most of the data in the real world are unlabeled, so learning the representation is particularly important for unsupervised learning task. In order to evaluate the effectiveness of our representation in unsupervised learning tasks, we design clustering experiment on three

Table 1: The result of clustering

	3-NG	6-NG	9-NG
DeepWalk	0.351	0.300	0.197
LINE	0.625	0.461	0.215
SDNE	0.423	0.365	0.254
GraphGAN	0.543	0.315	0.201
Struc2Vec	0.412	0.307	0.211
DNE	**0.709**	**0.527**	**0.425**

datasets: 3-NG, 6-NG and 9-NG, which are separated from 20-Newsgroup.

In the clustering task, we execute each baseline method to generate representations for the nodes, which are used as features for clustering. Then we cluster the nodes into categories with K-Means algorithm. And we record the performance with NMI (Normalized Mutual Information) score. The results of clustering are shown in Table 1

The results show that our method outperforms the others. Methods, such as DeepWalk and GraphGAN, only consider whether two nodes are connected and do not take the weight of edges into account. Therefore, these baselines are not applicable to the weighted dense networks. However, our method overcomes these obstacles. The proposed DW-random walk method not only considers the connection between two nodes, but also the weights of edges and the degrees of nodes. In combination with LSTM and LapEO, we can better preserve the network's global and local information. Therefore DNE is robust in the clustering task both on a weighted and unweighted dense network.

Visualization In visualization task, we focus on using the learned representation to reveal the network data intuitively. We execute our model and baseline methods on 3-NG dataset which comes from the 20-Newsgroup dataset. This dataset has 600 nodes each of which belongs to one of three categories which are $comp.graphics$, $rec.sport.baseball$ and $talk.politics.guns$. We map the representations learned by different network embedding methods into the 2-D space using the visualization tool $t\text{-}SNE$(Van, Hinton, and Maaten 2017). Figure 3 shows the visualization results on 3-NG dataset. Each point represents a document and colors indicate different categories. The visualizations of DeepWalk, Struc2Vec and GraphGAN is not meaningful, where the documents belonging to the same categories are not clustered together. For example, DeepWalk and Struc2Vec make the points belonging to different categories mix with each other. GraphGAN overlaps the nodes of different categories with each other. For LINE and SDNE, although the data can generally be divided into three clusters, the boundary is not clear enough. Obviously, the visualization of our method DNE performs better than baselines. This experiment demonstrates DNE model can learn more meaningful and robust representations.

Classification We divide the classification into two categories according to the number of labels in datasets that nodes have: multi-label classification which the node has more than one labels and multiclass classification which the node only has one definite label.In the multi-label classification task, i.e., BolgCatalog, every blogger author is assigned one or more interested topics as labels. The training phase with learn a classifier with a certain fraction of nodes and all their labels. The test task is to predict the labels for the remaining nodes. In multi-label classification experiments, we randomly sample a portion (from 10% to 90%) of the labeled nodes, and use them as training data. The rest of the nodes are used as test data. We report the average performance in terms of $Micro\text{-}F_1$ and $Macro\text{-}F_1$. The results are shown in Tabel 2. As we can see that although the performance of DeepWalk is very competitive, we still outperform it. It demonstrates that LSTM deep learning model more powerful than Skip-Gram model. Our model achieves most gains of $27.57\%(Micro\text{-}F_1)$ and $24.26\%(Macro\text{-}F_1)$ with 90% train nodes. That illustrate that the learned network representations of DNE can better generalize to the multilabel classification task than baselines.

In single-label node classification task, we employ Cora, Citeseer and Pubmed datasets. And we record the performance in terms of $Accuracy$. For classification experiment, we execute our model and baseline methods on the whole network to generate nodes representations for a one-vs-rest logistic regression classifier. We randomly sample 10% to 90% of the nodes as the training samples and use the left nodes to test the performance. The results are shown in Tabel 3, 4, and 5 respectively. For the PubMed dataset, we cannot get a result for the GraphGAN method with the current computational resources. From the results, it is evident we can see DNE achieves promising performance compared with others. For example, on CiteSeer, our method gives about 2% gain in accuracy over DeepWalk (ranked in the second place) under all training ratio settings.

Parameter Sensitivity

In this section, we will investigate the parameter sensitivity, w.r.t. Walks per node γ, Walk length l and Expected representation dimension d, in order to guide us in selecting the optimal parameters . We conduct the node classification task on Cora with 5:5 train-test ratio and use the $Accuracy$ to illustrate the performance of our model. We examine each parameter by fixing the other two parameters.

Figure 4a shows the results about dimension d. As the dimension increases, the accuracy first rises rapidly and then slowly declines. The number 128 is the most appropriate dimension. Figure 4b reveals the effect of γ and l on model's performance. The accuracy score is positively related to γ : as γ increases, the accuracy score rises linearly. γ corresponds to the capacity of the corpora, and the bigger γ is, the richer the corpora is. The length l of sequences generated by DW-Random Walk affects the global information of the node. LSTM can deal with long-term dependence sequence problems, so an increase in sequence length does not diminish the performance of the model.

Performance w.r.t. Random Walks

In this section we compared the performance of three types of Random Walks on the results, i.e., DW-$RandomWalk$, Truncated-$RandomWalk$, and Biased-$RandomWalk$. We

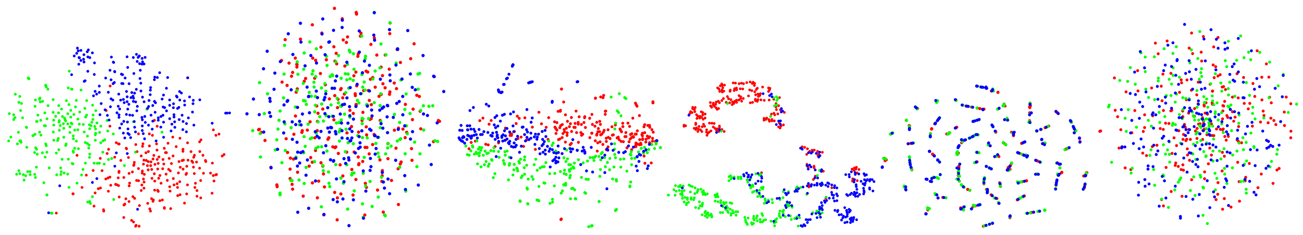


Figure 3: Visualization of 3-NG dataset. Each point represents one document. Different colors correspond to different categories, i.e., Red: $comp.graphics$, Blue: $rec.sport.baseball$, Green: $talk.politics.guns$

Table 2: The result of Multilabel classification on BlogCatalog

	% Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
	DeepWalk	33.12	36.20	37.60	39.30	**40.00**	40.30	40.50	**41.50**	42.00
	LINE	31.03	33.41	34.56	35.42	35.97	36.18	36.74	36.82	36.89
Micro-F1(%)	SDNE	30.81	31.94	32.56	32.98	33.12	33.31	33.47	33.57	33.95
	GraphGAN	23.43	24.83	25.30	25.62	25.67	25.51	25.69	25.73	25.34
	Struc2Vec	10.73	11.63	12.57	13.24	13.86	14.25	14.56	14.87	14.56
	DNE	**34.30**	**37.13**	**37.91**	**39.39**	39.97	**40.66**	**40.88**	41.32	**42.13**
	DeepWalk	**17.79**	20.02	21.62	22.57	23.18	24.66	24.73	25.11	27.37
	LINE	11.97	14.62	16.00	17.22	18.20	18.92	19.43	19.95	20.43
Macro-F1(%)	SDNE	14.94	15.55	16.03	16.11	16.31	16.41	16.54	16.69	16.87
	GraphGAN	9.53	10.08	10.32	10.44	10.38	10.27	10.44	10.44	9.88
	Struc2Vec	5.13	5.19	5.09	5.07	5.00	4.67	4.53	4.31	4.11
	DNE	17.52	**21.54**	**22.88**	**23.58**	**24.32**	**24.90**	**25.49**	**25.64**	**28.37**

Table 3: The result of Node Classification on Cora

	% Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
	DeepWalk	71.77	73.85	74.98	**76.12**	**77.65**	77.54	**78.84**	79.17	80.40
	LINE	46.02	51.18	54.00	55.22	56.38	56.89	57.24	57.97	59.29
Accuracy(%)	SDNE	38.89	39.30	39.26	38.85	38.72	38.64	38.92	38.70	38.67
	GraphGAN	24.58	26.64	27.44	27.75	28.72	29.33	29.42	29.62	28.71
	Struc2Vec	31.67	34.19	35.87	36.91	38.20	38.53	39.59	40.40	40.33
	DNE	**71.86**	**74.38**	**75.12**	76.06	77.03	**78.87**	78.22	**79.89**	**81.18**

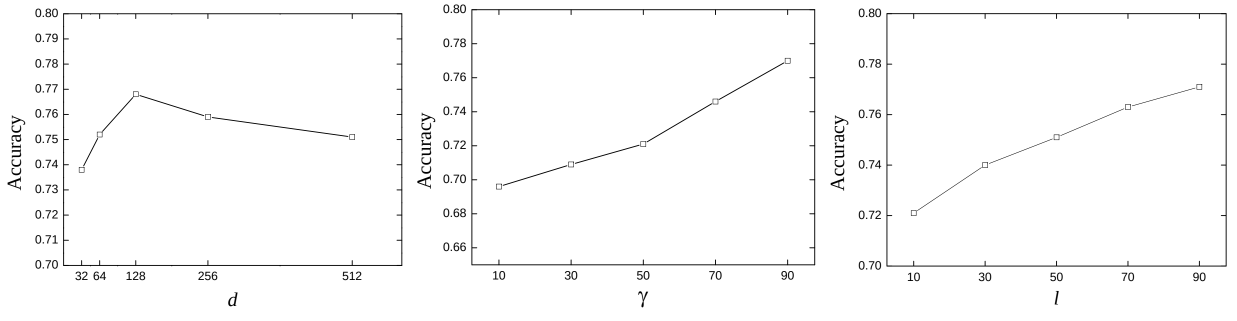
Table 4: The result of Node Classification on Citeseer

	% Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
	DeepWalk	50.16	52.42	54.53	55.53	56.12	57.02	57.68	58.63	59.42
	LINE	32.75	35.29	36.84	37.72	38.12	38.65	39.30	38.98	41.10
Accuracy(%)	SDNE	31.27	32.90	33.56	33.53	33.63	33.94	34.34	34.09	34.77
	GraphGAN	19.89	20.74	20.96	21.16	21.24	21.60	21.53	21.39	22.46
	Struc2Vec	25.92	26.94	27.99	28.56	28.50	29.64	30.26	30.33	31.92
	DNE	**51.72**	**54.32**	**55.47**	**56.08**	**57.15**	**58.22**	**59.75**	**59.45**	**61.56**

sample the node sequences on the BlogCatalog dataset with the three different Random Walk methods, and then used the DNE model to learn the representation. Figure 5 shows the results. We can see that our DW-$RandomWalk$ method outperforms the others. Biased-$RandomWalk$ is lightly better than Truncated-$RandomWalk$. Especially in case the number of labeled nodes is small, the performance of DW-$RandomWalk$ method is significantly better than the others. In fact, when trained with only 20% of the nodes labeled, DW-$RandomWalk$ performs better than Truncated-$RandomWalk$ and Biased-$RandomWalk$. It can be seen that considering the properties (such as weight and degree) of the network during random walks helps improve the performance of the model. DW-$RandomWalk$ utilizes both two important properties of the network, i.e., the degree of the node and weight of the edge, which allows exploring

Table 5: The result of Node Classification on Pubmed

	% Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
Accuracy(%)	DeepWalk	70.75	72.29	73.47	73.59	74.58	75.71	75.69	75.83	76.65
	LINE	53.49	54.43	54.96	55.19	55.43	55.56	55.68	55.39	55.18
	SDNE	39.57	40.02	40.52	40.45	41.20	41.58	41.64	41.56	41.69
	GraphGAN	-	-	-	-	-	-	-	-	-
	Struc2Vec	47.84	49.12	49.61	49.85	49.93	50.06	50.16	50.38	50.73
	DNE	**73.74**	**74.45**	**75.40**	**75.42**	**75.57**	**76.09**	**76.43**	**76.70**	**77.23**



(a) Dimension (b) Walks per node (c) Walk length

Figure 4: Parameter sensitivity analysis of our model on Cora with train ratio as 50%

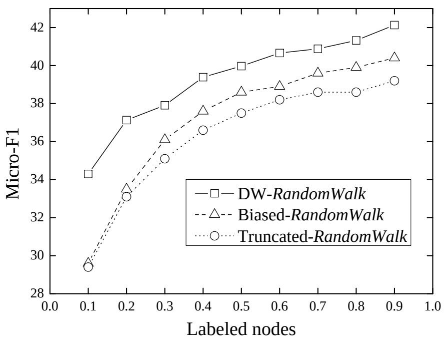


Figure 5: The results of different Random Walks

the functional role of nodes. The degree of the node can often reflect its importance in the network. By considering the degrees and weights jointly, we can traverse more diverse neighbor nodes And by introducing degree and weight information, we also avoid other hyper-parameters, such as p and q for the Biased-$RandomWalk$.

Discussion and Conclusion

Network structured data poses a great challenge for the traditional machine learning algorithms. Recent years researchers pay attention to designing feature vectors for learning algorithms. Structural function and neighborhood context are potential knowledge of the network structured data. And information transferring characterizes the structural roles of the nodes and connections, also the network evolution behaviors. This work introduces embedding layers to the traditional LSTM model to learn the representation for each network node. To the best of our knowledge, it is the first time of employing a prediction model to generate new node representations.

Random walk shows its nice ability on capturing the global neighborhood context in the former researches, such as Deepwalk (Perozzi, Al-Rfou, and Skiena 2014) and node2vec (Grover and Leskovec 2016). This paper further suggests a degree-weight biased random walk model to capture the functional and global network context information. In the network domain, the social role of the node is mostly dependent on the degree. And the weight denotes the relationships among nodes. The new walk stage can guide the walking process to capture the major social role and global structural context. The prediction model is mainly used to embed the network structure properties. Besides, it can also preserve the transfer possibilities among the nodes. To capture the local network structure, we propose a Laplacian supervised embedding space optimization method following the embedding layer of the model. Laplacian Eigenmaps ensure that two connected nodes are close in low-dimensional space, so it can extract local information from the network. Our proposed DNE method could generate a robust representation. Experiments on multiple datasets are conducted to evaluate the network representation generated by the method. The results show that our method is a valuable complement to the state-of-the-art.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments. This work was supported in part by the National Natural Science Foundation of China (No. 41741007, 41576011, U1706218).

References

Belkin, M., and Niyogi, P. 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, 585–591.

Bruna, J.; Zaremba, W.; Szlam, A.; and Lecun, Y. 2013. Spectral networks and locally connected networks on graphs. *Computer Science*.

Cao, S.; Lu, W.; and Xu, Q. 2016. Deep neural networks for learning graph representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 1145–1152. AAAI Press.

Dai, Q.; Li, Q.; Tang, J.; and Wang, D. 2017. Adversarial network embedding.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial networks. *Advances in Neural Information Processing Systems* 3:2672–2680.

Graves, A. 2013. Generating sequences with recurrent neural networks. *Computer Science*.

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

Henderson, K.; Gallagher, B.; Li, L.; Akoglu, L.; Eliassi-Rad, T.; Tong, H.; and Faloutsos, C. 2011. It's who you know: graph mining using recursive structural features. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 663–671. ACM.

Henderson, K.; Gallagher, B.; Eliassi-Rad, T.; Tong, H.; Basu, S.; Akoglu, L.; Koutra, D.; Faloutsos, C.; and Li, L. 2012. Rolx: structural role extraction and mining in large graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1231–1239. ACM.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *Computer Science*.

Kipf, T. N., and Welling, M. 2016. Semi-supervised classification with graph convolutional networks.

Lang, K. 1995. Newsweeder: learning to filter netnews. In *Twelfth International Conference on International Conference on Machine Learning*, 331–339.

Liben-Nowell, D., and Kleinberg, J. 2007. The link-prediction problem for social networks. *Journal of the American society for information science and technology* 58(7):1019–1031.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*.

Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning convolutional neural networks for graphs. 2014–2023.

Ou, M.; Cui, P.; Pei, J.; Zhang, Z.; and Zhu, W. 2016. Asymmetric transitivity preserving graph embedding. In *Proc. of ACM SIGKDD*, 1105–1114.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710. ACM.

Ribeiro, L. F. R.; Saverese, P. H. P.; and Figueiredo, D. R. 2017. struc2vec:learning node representations from structural identity. 385–394.

Roweis, S. T., and Saul, L. K. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science* 290(5500):2323–2326.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine* 29(3):93.

Sutskever, I.; Martens, J.; and Hinton, G. E. 2016. Generating text with recurrent neural networks. In *International Conference on Machine Learning, ICML 2011, Bellevue, Washington, Usa, June 28 - July*, 1017–1024.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. 4:3104–3112.

Tang, L., and Liu, H. 2009. Relational learning via latent social dimensions. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July*, 817–826.

Tang, J., and Liu, H. 2012. Feature selection with linked data in social media. In *12th SIAM International Conference on Data Mining (SDM2012)*.

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

Tenenbaum, J. B.; De Silva, V.; and Langford, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *science* 290(5500):2319–2323.

Tian, F.; Gao, B.; Cui, Q.; Chen, E.; and Liu, T. Y. 2014. Learning deep representations for graph clustering. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 1293–1299.

Van, der Maaten, L.; Hinton, G.; and Maaten, L. V. D. 2017. Visualizing data using t-sne. *Journal of Machine Learning Research* 9(2605):2579–2605.

Wang, H.; Wang, J.; Wang, J.; Zhao, M.; Zhang, W.; Zhang, F.; Xie, X.; and Guo, M. 2017. Graphgan: Graph representation learning with generative adversarial nets.

Wang, D.; Cui, P.; and Zhu, W. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1225–1234. 2939753: ACM.

Williams, R. J., and Peng, J. 1990. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation* 2(4):490–501.