Semi-Supervised Network Embedding

Conference Paper *in* Lecture Notes in Computer Science · March 2017

DOI: 10.1007/978-3-319-55753-3_9

Chaozhuo Li
Beihang University (BUAA)
24 PUBLICATIONS **129** CITATIONS
SEE PROFILE

Senzhang Wang
Nanjing University of Aeronautics & Astronautics
107 PUBLICATIONS **773** CITATIONS
SEE PROFILE

Yang Yang
Beihang University (BUAA)
17 PUBLICATIONS **156** CITATIONS
SEE PROFILE

Xiaoming Zhang
Beihang University (BUAA)
63 PUBLICATIONS **479** CITATIONS
SEE PROFILE

Semi-Supervised Network Embedding

Chaozhuo Li[1(✉)], Zhoujun Li[1], Senzhang Wang[2,3], Yang Yang[1], Xiaoming Zhang[1], and Jianshe Zhou[4]

[1] State Key Lab of Software Development Environment, Beihang University, Beijing, China
{lichaozhuo,lizj,yangyang}@buaa.edu.cn

[2] Nanjing University of Aeronautics and Astronautics, Nanjing, China
szwang@nuaa.edu.cn

[3] Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing, China

[4] Capital Normal University, Beijing 100048, People's Republic of China
yolixs@buaa.edu.cn

Abstract. Network embedding aims to learn a distributed representation vector for each node in a network, which is fundamental to support many data mining and machine learning tasks such as node classification, link prediction, and social recommendation. Current popular network embedding methods normally first transform the network into a set of node sequences, and then input them into an unsupervised feature learning model to generate a distributed representation vector for each node as the output. The first limitation of existing methods is that the node orders in node sequences are ignored. As a result some topological structure information encoded in the node orders cannot be effectively captured by such order-insensitive embedding methods. Second, given a particular machine learning task, some annotation data can be available. Existing network embedding methods are unsupervised and are not effective to incorporate the annotation data to learn better representation vectors. In this paper, we propose an order sensitive semi-supervised framework for network embedding. Specifically, we first propose an novel order sensitive network embedding method: StructuredNE to integrate node order information into the embedding process in an unsupervised manner. Then based on the annotation data, we further propose an semi-supervised framework SemNE to modify the representation vectors learned by StructuredNE to make them better fit the annotation data. We thoroughly evaluate our framework through three data mining tasks (multi-label classification, network reconstruction and link prediction) on three datasets. Experimental results show the effectiveness of the proposed framework.

1 Introduction

Curse of dimensionality is a fundamental problem that makes many learning tasks difficult. This issue is particularly obvious when one wants to model the joint distribution among many discrete random variables, such as the words in

DOI: 10.1007/978-3-319-55753-3_9

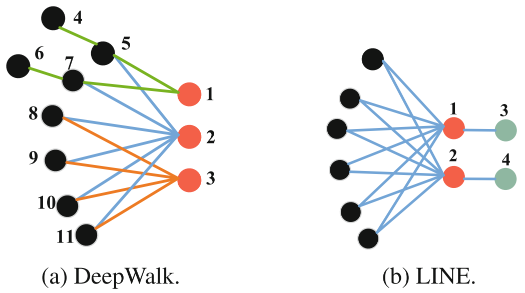


Fig. 1. Two illustrations of DeepWalk and LINE.

a sentence and the nodes in a sparse network. To address this issue, recently a large body of works [1,14,17,21,25] focused on investigating how to represent the node in a sparse network as a dense, continuous, and low-dimensional vector. Such a distributed vector is defined as the representation vector of each node and the learning of the representation vectors can be also referred to *network embedding* [17]. As a more promising way to represent nodes in the network, network embedding is the foundation for many machine learning and data mining tasks, such as node classification [3,5], link prediction [11], network reconstruction [21,23] and recommendation [24].

Recently, some proven neural network techniques are introduced to perform network embedding [7,9,14,21]. Neural network based methods are easier to generalize and can usually achieve more promising embedding performance, and thus have attracted rising research interests recently. Although several network embedding methods are proposed as mentioned above, it is still very difficult for them to learn promising representation vectors which can both effectively preserve the network structure information and fit a particular data mining task. In general, there are two major limitations for existing methods. Firstly, existing methods [14,25] directly apply order insensitive embedding methods to generate node representations, while the node order information in the node sequences is largely ignored. DeepWalk assumes that two nodes with similar contextual nodes in the node sequences tend to be similar, and the contextual nodes of a center node are defined as a fixed-size window of its previous nodes and after nodes in the sequences. As shown in Fig. 1(a), node 1 and 2 share common contextual nodes {4,5,6,7}, while node 2 and 3 share common contextual nodes {8,9,10,11}. DeepWalk considers that the similarity between nodes 1 and 2 should be close to the similarity between nodes 2 and 3, since the nodes in each pair share equal number of common contextual nodes. However, according to literature [6], the influence of the direct neighbor nodes on a specific node can be significantly larger than those that are farther from it. Hence in Fig. 1(a), the similarity between nodes 2 and 3 should be larger than that between nodes 2 and 1 because the common contextual nodes of nodes 2 and 3 are all their direct neighbors. LINE [17] and SDNE [21] both utilize first-order and second-order proximity to conduct network embedding. As shown in Fig. 1(b), according to the first-order proximity, nodes 1 and 3 are similar to each other because they are directly connected; according to second-order proximity, nodes 1 and 2 are similar to each other because they share many common neighbors. However, these two proximities only preserve limited global network structure information. As shown in Fig. 1(b), nodes 3 and 4 are not directly connected and share no neighbors, but they are also very likely to be similar due to the fact that their only neighbor nodes 1 and 2 are similar.

The second limitation is that existing network embedding methods are unsupervised. Given a particular machine learning task, a small number of manual annotation data might be available [4]. Previous unsupervised network embedding methods are not effective to integrate the annotation data into the embedding process. In such a case, directly utilizing a general purposed network embedding method and ignoring these labeled samples may not achieve desirable performance for such a particular task.

In this paper, we study the order sensitive semi-supervised network embedding problem. The studied problem is difficult to address due to the following three challenges. Firstly, the objective function of previous embedding methods cannot effectively capture the node order information, and thus a new order-sensitive embedding method is necessary. Secondly, due to the expensive cost of manual annotation, in many scenarios there are only a few labeled nodes available. Consider the scarcity of labeled data, it is challenging to propagate the label information from the labeled nodes to the vast unlabeled ones. Finally, since the network topology and the annotation data potentially encode different types of information, combining them is expected to give a better performance. However, it is not obvious how best to do this in general.

To address the above challenges, we propose a two-stage semi-supervised network embedding framework. In the first stage, we learn an initial representation vector for each node in an unsupervised way. To effectively capture the node order information, we propose StructuredNE to encode the node order information into network embedding by exploiting an novel order-sensitive Skip-Gram algorithm. In the second stage, to incorporate a small number of labeled samples, we further propose a semi-supervised network embedding framework SemNE to modify the initial node representations learned from the first stage. Based on the observation that center node and its contextual nodes are likely to share the same label, we try to modify the representation vectors of the contextual nodes according to the label of the center node.
To summarize, we make the following primary contributions:

- We show that the order information in a sampled node sequence can affect the performance of network embedding, then we propose a novel model to incorporate the node order information into the embedding process.
- We propose a novel neural network based semi-supervised learning schema to integrate a small number of labeled data into network embedding process.
- We extensively evaluate our approach through multi-label classification, network reconstruction and link prediction tasks on three datasets. Experimental results show the effectiveness of the proposed embedding framework.

The rest of this paper is organized as follows. Section 2 formally defines the problem of semi-supervised network embedding. Section 3 introduces the proposed semi-supervised embedding framework in details. Section 4 presents the experimental results. Section 5 summarizes the related works. Finally we conclude this work in Sect. 6.

2 Problem Definition

A network G is defined as $G = (V, E)$, where V represents nodes in the network and E contains the edges. Existing unsupervised network embedding methods only utilize the network G. Different from previous works, in our setting we also have some labeled data, and we aim to fully utilize them to learn better node embeddings. The annotation matrix $Y \in \mathbb{R}^{|V_t| \times |\mathcal{Y}|}$ contains partial label information, in which $\mathcal{Y}$ is the set of label categories, and node set V_t contains a small number of labeled nodes. We formally define the problem of semi-supervised network embedding as below:

Definition 1. (Semi-supervised Network Embedding) *Given a network $G = (V, E)$ and the label matrix Y, the problem of* semi-supervised network embedding *aims to learn such a matrix $X \in \mathbb{R}^{|V| \times d}$, where d is the number of latent dimensions with $d \ll |V|$, and each row vector X_i of X is the representation vector of node i. We want to make the node similarity calculated by their representation vectors preserves both the network structure and manual annotation information.*

3 Semi-supervised Network Embedding Framework

In this section, we present the details of the semi-supervised network embedding framework. Firstly we introduce the frame diagram of the proposed embedding framework. Then DeepWalk, the basis of our work, is briefly introduced. After that, we present two critical components of the proposed framework: the order sensitive unsupervised embedding model StructuredNE and the semi-supervised embedding model SemNE.

3.1 Framework

As shown in Fig. 2, our network embedding framework contains two stages. In the first stage, the network is transformed into a set of node sequences using random walk. Based on these sequences, the order sensitive embedding model StructuredNE is introduced to perform network embedding. The learned representation vectors from the first stage preserve the network topology and node order information. In the second stage, based on the annotation data, we further propose an semi-supervised network embedding framework SemNE to modify the initial representations learned from the first stage. In order to transform the annotation information from labeled nodes to unlabeled ones, we generate a set of training units and their labels. Different colors of the training units represent

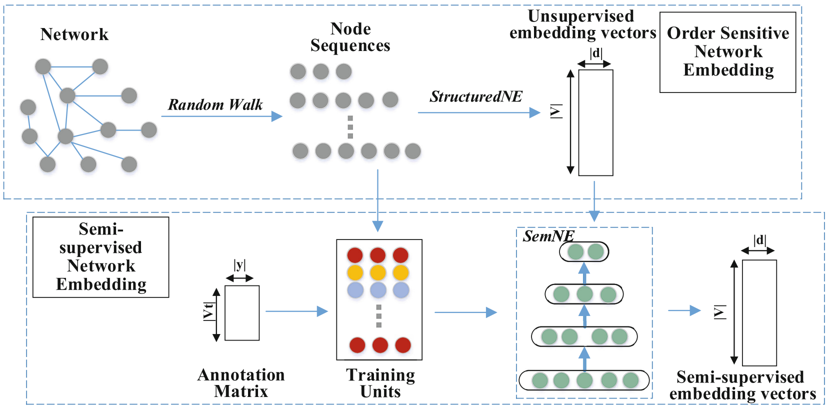

Network
Node Sequences
Random Walk
StructuredNE
Unsupervised embedding vectors
|d|
|V|
Order Sensitive Network Embedding
Semi-supervised Network Embedding
|y|
|Vt|
Annotation Matrix
Training Units
SemNE
Semi-supervised embedding vectors

Fig. 2. Framework of the semi-supervised network embedding model.

different labels. A labeled center node together with its contextual nodes in a window are treated as a training unit, and we consider the label of the training unit to be the same as the center node. We propose a neural network model to predict the labels of the training units, and with error back propagation algorithm, the original node representation vectors are modified to match the labels of training units.

3.2 DeepWalk

In this subsection, we briefly present DeepWalk [14], the basis of our work. DeepWalk introduced Skip-Gram algorithm [12,13] into the study of social network to learn the representation vector for each node. Firstly, DeepWalk generates a set of node sequences by random walk. Given a node sequence $T = \{n_1, n_2, ..., n_{|T|}\}$, nodes $n \in \{n_{t-w}, ..., n_{t+w}\} \backslash n_t$ are regarded as the contextual nodes of the center node n_t, where w is the window size. The objective function of the Skip-Gram model is to maximize the likelihood of the prediction of contextual nodes given the center node. More formally, given a node sequence $T = \{n_1, n_2, ..., n_{|T|}\}$, Skip-Gram aims to maximize the following objective function:

$$L = \frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-w \leq j \leq w \\ j \neq 0}} \log p(n_{t+j}|n_t) \tag{1}$$

In order to obtain the output probability $p(n_{t+j}|n_t)$, the Skip-Gram model estimates a matrix $O \in \mathcal{R}^{|V| \times d}$, which maps the representation vector $X_{n_t} \in \mathcal{R}^{1 \times d}$ of node n_t, into a $|V|$-dimensional vector o_{n_t}:

$$o_{n_t} = X_{n_t} \cdot O^T \tag{2}$$

where d is the embedding dimension. Then, the probability of the node n_{t+j} given the node n_t can be calculated by the following softmax objective function:

$$p(n_{t+j}|n_t) = \frac{e^{o_{n_t}(n_{t+j})}}{\sum_{n \in V} e^{o_{n_t}(n)}} \tag{3}$$

In the network embedding process, the representation vector X_{n_t} is modified to maximize L. After the learning process, we can obtain a matrix $X \in \mathbb{R}^{|V| \times d}$ which contains the representation vectors of all nodes.

Figure 3(a) shows the framework of Skip-Gram model. Skip-Gram model uses each center node as an input to a log-linear classifier with continuous projection layer. The projection layer convert each input node to the corresponding representation vector, and the log-linear classifier predict the contextual nodes given the center node. From the objective function of Skip-Gram model, we can clearly see that the node order information is ignored. In Formula 3, the probability $P(n_{t+j}|n_t)$ is independent from the node order index j, which may leads to suboptimal results.

3.3 Order Sensitive Unsupervised Network Embedding

In this subsection we propose an order sensitive network embedding method StructuredNE, which introduces the structured Skip-Gram model [22] for order sensitive network embedding. By introducing changes that make the embedding process aware of the relative positioning of contextual nodes, our goal is to improve the learned representation vectors while maintaining the simplicity and efficiency.

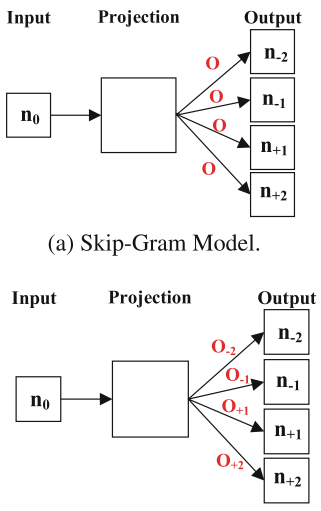


Fig. 3. Skip-Gram and Structured Skip-Gram Model.

Algorithm 1: StructuredNE

```
    Input:
1        $G(V, E)$ ;
2        window size w;
3        embedding size d;
4        walks per node r;
5        walk length t;
    Output:
6        $X \in \mathbb{R}^{|V| \times d}$ ;
7   $\mathcal{P} = [\,]$;
8   for $i = 0; i < r; i++$ do
9       $\mathcal{O}$ = Shuffle($V$);
10      for $v_i$ in $\mathcal{O}$ do
11          $path$ = RandomWalk($G$, $v_i$, $t$);
12          $\mathcal{P}$.append($path$);
13      end
14  end
15  $X$ = StructuredSkipGram($\mathcal{P}, w, d$);
16  return $X$;
```

Figure 3(b) shows the framework of the structured Skip-Gram model. Different from the original Skip-Gram model, structured Skip-Gram model defines a set of $w \times 2$ output predictors O_{-w}, ...,O_{-1},O_1,O_w, with each $O_i \in \mathcal{R}^{|V| \times d}$. w is the the window size of contextual nodes. Each of the output matrices is dedicated to predicting the output for a specific relative position to the center node. When making a prediction $p(n_j|n_i)$, we select the appropriate output matrix O_{j-i} to project the network embeddings to the output vector. More formally, given a node sequence $T = \{n_1, n_2, ..., n_{|T|}\}$ in $\mathcal{P}$, the structured Skip-Gram model aims to maximize:

$$L = \frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-w \leq j \leq w \\ j \neq 0}} \log p(n_{t+j}|n_t) \tag{4}$$

in which

$$p(n_{t+j}|n_t) = \frac{e^{o_{j,n_t}(n_{t+j})}}{\sum_{n \in V} e^{o_{j,n_t}(n)}} \tag{5}$$

o_{n_t} is a $|V|$-dimensional vector calculated by:

$$o_{j,n_t} = X_{n_t} \cdot O_j^T \tag{6}$$

One can see that the objective function of structured Skip-Gram model is affected by the relative positions between the center node and its contextual nodes. Hence, order information in the node sequences is integrated into the embedding process.

Algorithm 1 shows the main steps of the proposed StructuredNE. Lines 7 to 14 reveal the generation process of node sequences. In an iteration from line 9 to 13, after shuffling all nodes in V, the random walk generator function $RandomWalk(G, v_i, t)$ generates a single node sequence. In network G, starting from node v_i, a node sequence of length t is generated by random walk. This kind of iteration will perform r times to obtain enough sequences. All node sequences are appended into list $\mathcal{P}$. Then, the structured Skip-Gram model is introduced to learn the node representations with dimension d, and w represents the window size of contextual nodes.

Note that, the proposed order-sensitive embedding model does not increase the time cost. The number of calculations that must be performed for the forward and backward passes in the learning process of StructuredNE remains the same to the DeepWalk, as we are simply switching the output layer O for each different relative node positions. The number of parameters in matrix O is linearly increased by a factor of $w \times 2$.

3.4 Semi-supervised Network Embedding

In this subsection we propose an semi-supervised network embedding framework SemNE to incorporate the annotation data. In NLP area, many kinds of neural

Table 1. Generation of training units

(a) Example of node sequence and categories	
Node sequence	$\{n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9\}$
Categories	n_3 : category 1
	n_5 : category 2
	n_7 : category 3
(b) Training units extracted from above data with window size as 2	
Training unit	Label
$\{n_1, n_2, n_3, n_4, n_5\}$	category 1
$\{n_3, n_4, n_5, n_6, n_7\}$	category 2
$\{n_5, n_6, n_7, n_8, n_9\}$	category 3

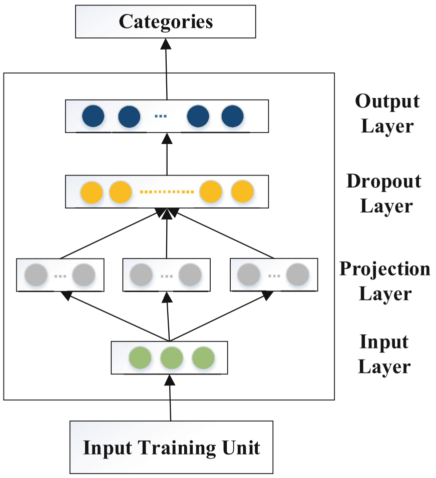


Fig. 4. Framework of SemNE.

networks have been proposed to process the corresponding tasks, such as the log-linear networks [12] and deeper feed-forward neural networks [8,16]. Motivated by these works, we modify the embedding vectors obtained from unsupervised methods by an neural network classification model to obtain the semi-supervised network embedding results.

The insight of SemNE is to utilize the manual annotations to tune the initial representation vectors by a neural network learning framework. The initial representation vectors of nodes are learned by the unsupervised embedding methods. In order to transform the annotation information from labeled nodes to unlabeled ones, we propose to utilize the training unit as the input of SemNE. Assume node n_i is a labeled node in V_t, given a node sequence contains n_i, the combination of n_i and its contextual nodes in a fixed window size w are treated as a training unit, and the label of the training unit is the same as the center node n_i. Table 1 shows an example of training units. Table 1(a) contains a node sequence and some labeled nodes, and Table 1(b) shows training units and their labels generated from the data of Table 1(a). We try to transfer annotation information from the center node to its contextual nodes in the same unit, and jointly learn their semi-supervised embeddings. As we sample a large number of node sequences for learning, thus statistically the unlabeled nodes in a training unit have a higher probability to be assigned the same label to many its neighbors sharing the same label, while a lower probability to be assigned other labels otherwise.

Given a training unit u, our neural network model tries to learn its distribution on different classes $P(\mathcal{Y}_i|u)$. As shown in Fig. 4, SemNE includes following layers:

- **Input Layer**: A training unit is submitted into the input layer. Given the window size w as 2, assume the input training unit is $\{n_{-2}, n_{-1}, n_0, n_1, n_2\}$, where $\{ n_{-2}, n_{-1}, n_1, n_2\}$ are the contextual nodes of n_0. The label of the training unit is as same as the label of n_0. Input layer maps nodes in the input training unit to their corresponding indexes. The output vector of this layer is a w-dimensional vector containing the indexes of nodes in the input training unit.
- **Projection Layer**: This layer maps the input w-dimensional vector to a $(w \cdot d)$-dimensional vector. According to the input node indexes, project layer looks up their corresponding representation vectors in the embedding matrix X, which is initialized by unsupervised network embedding methods. After that, these representation vectors are concatenated as the output vector of project layer.
- **Dropout Layer**: In order to avoid overfitting, we add a dropout layer to the network to make use of its regularization effect. During training, the dropout layer copies the input to the output but randomly sets some of the entries to zero with a probability, which is set to 0.5.
- **Output Layer**: This layer linearly maps the input vector $I \in \mathbb{R}^{1\times(w\cdot d)}$ into a $|\mathcal{Y}|$-dimensional vector $\mathcal{O}$:

  $$\mathcal{O} = W \cdot I^T + b; \tag{7}$$

  where $W \in \mathbb{R}^{|\mathcal{Y}|\times(w\cdot d)}$ and $b \in \mathbb{R}^{|\mathcal{Y}|\times 1}$ are parameters to be learned. $\mathcal{Y}$ is the set of categories. Then softmax function is introduced as the activation function:

  $$P(\mathcal{Y}_i|u) = \frac{e^{\mathcal{O}_i}}{\sum_{0\le k\le|\mathcal{Y}|-1} e^{\mathcal{O}_k}} \tag{8}$$

  The output of this layer can be interpreted as a conditional probability over categories given the input unit.

The entire formulation of our framework is:

$$\begin{gathered}\mathcal{O} = W \cdot DropOut(Concat_{0\le i\le w-1}(X_{u_i})) + b;\\ P(\mathcal{Y}_i|u) = \frac{e^{\mathcal{O}_i}}{\sum_{0\le k\le|\mathcal{Y}|-1} e^{\mathcal{O}_k}}\end{gathered} \tag{9}$$

where u is the input training unit and w denotes the window size. $X \in \mathbb{R}^{|V|\times d}$ contains the pre-trained node representations. *Concat* is the concatenation function used in project layer. *DropOut* denotes the dropout layer. W and b are parameters in the output layer. We utilize SGD and error back propagation to minimize negative log-likelihood cost function for each training example (u, l). The loss function is defined as

$$Loss(u,l) = -\log\left(\frac{e^{\mathcal{O}_l}}{\sum_{0\le k\le|\mathcal{Y}|-1} e^{\mathcal{O}_k}}\right) = -\mathcal{O}_l + \log\left(\sum_{0\le k\le|\mathcal{Y}|-1} e^{\mathcal{O}_k}\right) \tag{10}$$

During the training process, parameters in the neural network are modified to minimize the loss function. The embedding matrix X, which is used as parameters in the project layer, is also modified to adapt the annotation data. After the training process, the modified matrix X contains the semi-supervised representation vector for each node.

4 Experiments

In this section, firstly we introduce the datasets and comparison methods used in our experiments. Then we evaluate the proposed embedding methods through three data mining tasks on three social network datasets. Also we discuss the experimental results and investigate the sensitivity across core parameters.

4.1 Experiment Setup

In this section, we introduce the setup of our experiments, including datasets, baseline methods and the parameter setup.

Datasets. We use the following network datasets for evaluation:

- **BlogCatalog**: In the BlogCatalog website, bloggers submit blogs and specify the categories of these blogs. Blogger can follow other bloggers and form a social network. Nodes are the bloggers and labels represent their interest categories.
- **Flickr**: Flickr is a popular website where users can upload photos. Besides, there are various groups that are interested in some specific subjects. Users can join several groups they are interested in. In this data, nodes are users and labels represent interest groups users join in. We select top-10 most popular interest groups as categories of this dataset.
- **YouTube**: YouTube is a popular video sharing website. Users can upload videos and follow other similar interest users, which forms the social network. In this data, nodes are users and labels represent the interest groups of the users.

Table 2 displays the detailed statistics of the three datasets used in our experiments.

Table 2. Statistics of the datasets

Data	BlogCatalog	Flickr	YouTube
Categories	39	10	47
Nodes	10, 312	80, 513	1, 138, 499
Links	333, 983	5, 899, 882	2, 990, 443

Embedding Methods. We compare the proposed embedding methods with state-of-the-art methods as follows, including both matrix factorization based methods and neural network based methods.

– **TruncatedSVD**: TruncatedSVD [20] performs linear dimensionality reduction through truncated singular value decomposition (SVD).
– **LINE**: LINE [17] is a state-of-the-art network embedding method. As shown in [17], we concatenate the first-order and second-order embedding results as the final representation vector.
– **DeepWalk**: DeepWalk [14] introduces the Skip-Gram embedding method into social representation learning.
– **StructuredNE**: StructuredNE is the proposed order sensitive embedding method.

To further evaluate the performance of various semi-supervised embedding methods, we conduct experiments with the following methods.

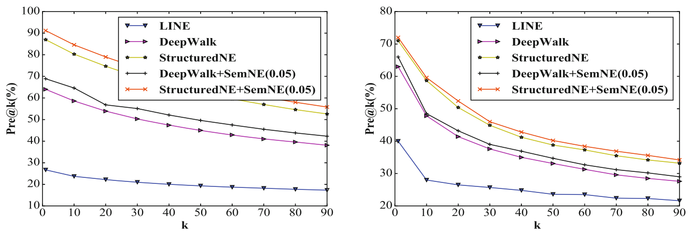
– **DeepWalk + SemNE(p)**: The embedding results of DeepWalk are treated as the initial representation vectors of nodes. Given p percents nodes as known labeled information, the initial vectors are modified by using SemNE model to generate the final representation vectors.
– **StructuredNE + SemNE(p)**: Similar to DeepWalk + SemNE(p), StructuredNE first learns the initial representation vector for each node, and then SemNE is utilized to generate the final representation vectors.

Parameter Setup. Following most network embedding methods [14,17], we set the dimension of representation vector to 128. In LINE method, the parameters are set as follows: negative = 5 and samples = 100 million. In DeepWalk methods, parameters are set as window size w = 5, walks per node r = 80 and walk length t = 40. In semi-supervised embedding model SemNE, the size of training unit w = 5 and the percentage of labeled node set V_t used in learning process p = 0.05.

4.2 Network Reconstruction

In this subsection, we represent the network reconstruction performance of different methods. Wang et al. [21] propose network reconstruction task as a basic evaluation of different embedding methods. A good network embedding method should ensures that the learned embedding vectors are able to reconstruct the original network.

Assume the node representation vectors has been generated by network embedding methods. Given a specific node in the network, we can calculate the similarity scores of representation vectors between it and other nodes, and sort other nodes on descending order of the similarity scores. The existing links in the original network are known and can serve as the ground-truth. Neighbor nodes of the specific node are regarded as positive samples and nonadjacent nodes are negative samples. We choose $pre@k$ to evaluate the number of positive samples appeared in top k similar nodes, which is defined as:



(a) BlogCatalog. (b) Flickr.

Fig. 5. Network reconstruction performance of *pre@k*.

$pre@k = \frac{|\{j|i,j \in V, index(j)k, E_{ij}=1\}|}{k}$, in which $index(j)$ is the ranked index of the j-th node and $E_{ij} = 1$ refers to node i and j are directly connected.

We conduct experiments with k from 10 to 90 and Fig. 5 presents the corresponding results. From the experimental results, one can tell that our proposed methods consistently outperform all the other baselines on two datasets. Among all unsupervised embedding methods(Line, DeepWalk, StructuredNE), StructuredNE achieves the best performance and beats DeepWalk by 20% in BlogCatalog and 10% in Flickr, which proves the order information is helpful to preserve the network structure information. LINE performs worst on this task because it directly concatenate first-order and second-order embedding vectors as the final vector. The direct linked relationships between nodes are ignored by the second-order embedding vectors, which will introduce noise into the similarity calculation. Furthermore, after introduced a few annotation data by SemNE model, the performance is further improved.

4.3 Multi-Label Classification

In this subsection, we provide and analyze the multi-label classification performance of different embedding methods. We design the following classification methods.

- **Embedding Based Classification:** This kind of classification methods utilize representation vectors generated by various network embedding methods in Sect. 4.1. The representation vector of each node is treated as its feature vector, and then we use a one-vs-rest logistic regression model to return the most likely labels.
- **SpectralClustering** [19]**:** SpectralClustering conducts a low-dimension embedding of the affinity matrix between samples, followed by a k-means algorithm.
- **EdgeCluster** [18]**:** This method uses k-means clustering to cluster the adjacency matrix of G, which can handle the huge graphs.

In the embedding based classification methods, the one-vs-rest logistic regression classifier is implemented using scikit-learn[1] tool. A portion (T_r) of the

[1] http://scikit-learn.org/stable/.

labeled nodes are randomly picked as the training samples, and the rest nodes are treated as the test samples. We repeat this process 10 times, and report the average performance in terms of both macro-F1 and micro-F1. In EdgeCluster and SpectralClustering methods, the parameters are set the same as reported in the literatures. Figure 6 shows the results of classification on the three datasets. When the percentage of training data is set to 0.2, the improvement over the baseline methods (DeepWalk, LINE) is statistically significant (sign test, p-value < 0.05).

Experimental Results and Analysis. For the BlogCatalog dataset, we conduct experiments with the training data ratio increasing from 10% to 90%. From the experimental results shown in Fig. 6(a), one can see that among the unsupervised embedding methods, by incorporating the node order information, StructuredNE achieves better performance than other baselines. One can also see that, by incorporating a small number of annotation data into the proposed SemNE model, the performance of semi-supervised methods increases nearly 5% compared to the corresponding unsupervised methods. In Flickr dataset, our proposed order sensitive embedding method StructuredNE performs better than other unsupervised methods. The proposed semi-supervised method SemNE further improves the classification performance. Overall, StructuredNE + SemNE achieves the best performance on this data set and beats the best baseline method DeepWalk by nearly 5%. YouTube dataset is much larger than the other two datasets. Some baseline methods such as TruncatedSVD and SpectralClustering are not applicable due to the huge size of the data. We increase the training data ratio (T_r) from 1% to 9%. Among unsupervised embedding methods, our proposed order sensitive embedding method StructuredNE outperforms DeepWalk. By utilizing the labeled data, SemNE model can further improve the micro-F1 and macro-F1 scores.

From the results on the three datasets, we can obtain the following conclusions. (1) The proposed order sensitive embedding method StructuredNE outperforms the order insensitive embedding method DeepWalk; (2) Compared to the unsupervised network embedding methods, the proposed semi-supervised embedding method SemNE does improve the performance, and the initial representation vectors in SemNE can affect the classification performance.

Parameter Sensitivity. As the percentage p of the available labels can remarkably affect the performance of SemNE, here we conduct experiments to study the effect of different p. We evaluate the classification performance on BlogCatalog and Flickr datasets with various p. As shown in Fig. 7, we can see that on both datasets, the micro-F1 and macro-F1 scores consistently increase with the increase of p. The results also show that the initial representation vectors can affect the classification performance of SemNE, because StructuredNE + SemNE consistently outperforms DeepWalk + SemNE.

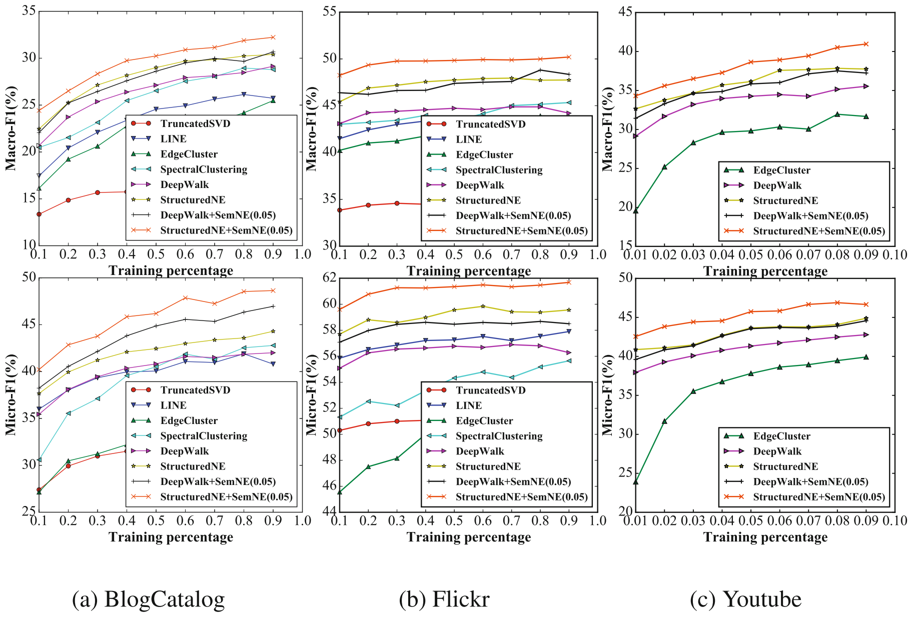


Fig. 6. Classification performance (Micro-F1 and Macro-F1) on three datasets.

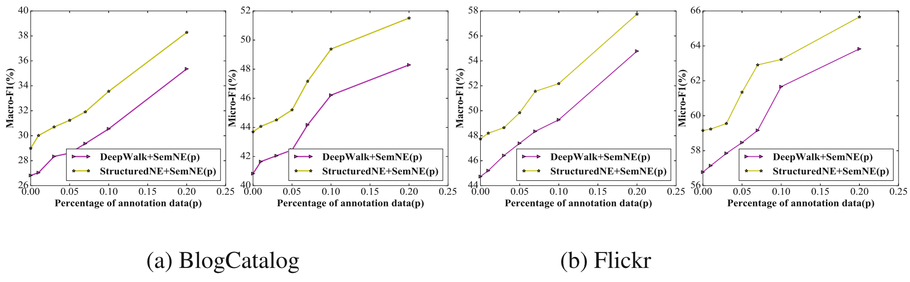


Fig. 7. The classification performance w.r.t the percentage p of labeled nodes.

4.4 Link Prediction

In this task, we wish to predict whether a pair of nodes should have an edge connecting them. Link prediction is useful in many domains, such as in social networks, it helps us predict friend relationships between users. Different from the network reconstruction, this task predicts the future links instead of reconstructing the existing links.

Firstly we remove a portion of existing links from the input network. According to the remaining subnetwork, we generate the representation vector for each node using different embedding methods. Node pairs in the removed edges are considered as the positive samples. Besides, we randomly sample an equal

Table 3. Link prediction performance (AU score)

Method	BlogCatalog	Flickr
LINE	0.693	0.526
DeepWalk	0.742	0.628
Common neighbors	0.654	0.502
StructuredNE	0.796	0.661
StructuredNE + SemNE(0.05)	**0.821**	**0.683**

number of node pairs which are not directly connected as the negative samples. The similarity score between two nodes in a sampled node pair are calculated according to their representation vectors. We choose the Area Under Curve (AUC) score to measure the consistency between labels and the similarity scores. In addition to the embedding based methods, we also choose an popular and proven method *Common Neighbor* [10].

Table 3 shows the experimental results of link prediction. After removed 40% edges from the input network, our proposed embedding method StructuredNE + SemNE outperforms other baseline methods by 8% in BlogCatalog and 6% in Flickr dataset, which prove our proposed embedding method is still effective in sparse networks.

5 Related Work

To address the network embedding problem, conventional matrix factorization based approaches first construct the affinity matrix from the input network, and then generate the representation vectors during the decomposition process of the affinity matrix. Locally linear embedding [15] generate the embedding vectors which preserve distances within local neighborhoods. Spectral Embedding [2] finds the non-linear embeddings of the input affinity matrix using a spectral decomposition of the Laplacian graph. However, previous methods rely on the factorization of the affinity matrix, the complexity of which is too high to apply on a large network.

Recently the neural network techniques are introduced to learn the representation vectors of nodes, which can counter the limitations of matrix factorization based approaches. DeepWalk [14] introduces an order-irrelevance word embedding algorithm (Skip-Gram) to perform network embedding. Tang et al. [17] propose two types of proximities to preserve the global and local topology information, and they further propose a general embedding model LINE to utilize these proximities. Wang et al. [21] propose a deep non-linear embedding model to capture the first-order and second-order proximities in the network. Node2vec [9] first defines the notion of network neighborhood, and then proposed an improved random walk procedure, which efficiently explores diverse neighborhoods. However, previous related works are unsupervised learning methods which only considers the network structure information. Different from previous works, given a few annotation data, our work aims to propose an semi-supervised learning framework which is able to incorporate these label information for better learning the embedding vectors.

6 Conclusion

This paper firstly proposes an order sensitive unsupervised embedding model StructuredNE. Then we propose an novel semi-supervised network embedding framework SemNE, which is a neural network based model to integrate the annotation information into network embedding process. Experimental results of multi-label classification, link prediction and network reconstruction tasks on three datasets demonstrate the effectiveness of the proposed methods.

Acknowledgement. This work was supported by Beijing Advanced Innovation Center for Imaging Technology (No.BAICIT-2016001), the National Natural Science Foundation of China (Grand Nos. 61370126, 61672081, 61602237, U1636211, U1636210), National High Technology Research and Development Program of China (No.2015AA016004), the Fund of the State Key Laboratory of Software Development Environment (No.SKLSDE-2015ZX-16).

References

1. Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V., Smola, A.J.: Distributed large-scale natural graph factorization. In: WWW, pp. 37–48 (2013)
2. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput. **15**(6), 1373–1396 (2003)
3. Bhagat, S., Cormode, G., Muthukrishnan, S.: Node classification in social networks. In: Social Network Data Analytics, pp. 115–148. Springer, US (2011)
4. Cao, J., Wang, S., Qiao, F., Wang, H., Wang, F., Yu, P.S.: User-guided large attributed graph clustering with multiple sparse annotations. In: Bailey, J., Khan, L., Washio, T., Dobbie, G., Huang, J.Z., Wang, R. (eds.) PAKDD 2016. LNCS (LNAI), vol. 9651, pp. 127–138. Springer, Cham (2016). doi:10.1007/978-3-319-31753-3_11
5. Cao, S., Lu, W., Grarep, Q.: Learning graph representations with global structural information. In: CIKM, pp. 891–900. ACM, New York (2015)
6. Cha, M., Haddadi, H., Benevenuto, F., Gummadi, P.K.: Measuring user influence in twitter: the million follower fallacy. In: ICWSM 2010 (2010)
7. Chang, S., Han, W., Tang, J., Qi, G.-J., Aggarwal, C.C., Huang, T.S.: Heterogeneous network embedding via deep architectures. In: KDD, pp. 119–128. ACM, New York (2015)
8. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. JMLR **12**, 2493–2537 (2011)
9. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: KDD (2016)
10. Liben-Nowell, D., Kleinberg, J.: The link prediction problem for social networks. In: CIKM, pp. 1019–1031 (2003)
11. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. J. Am. Soc. Inf. Sci. Technol. **58**(7), 1019–1031 (2007)

12. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
13. Morin, F., Bengio, Y.: Hierarchical probabilistic neural network language model. In: Aistats, vol. 5, pp. 246–252. Citeseer (2005)
14. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: KDD, pp. 701–710. ACM, New York (2014)
15. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–2326 (2000)
16. Taghipour, K., Ng, H.T.: Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In: NAACL, pp. 314–323 (2015)
17. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: WWW, pp. 1067–1077 (2015)
18. Tang, L., Liu, H.: Leveraging social media networks for classification. In: CIKM, pp. 1107–1116 (2009)
19. Tang, L., Liu, H.: Leveraging social media networks for classification. Data Min. Knowl. Disc. **23**(3), 447–478 (2011)
20. Tropp, A., Halko, N., Martinsson, P.: Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. Technical report (2009)
21. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: KDD (2016)
22. Wang, L., Dyer, C., Black, A., Trancoso, I.: Two/too simple adaptations of word2vec for syntax problems. In: NAACL, pp. 1299–1304 (2015)
23. Wang, S., Hu, X., Yu, P.S., Li, Z.: Mmrate: inferring multi-aspect diffusion networks with multi-pattern cascades. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1246–1255 (2014)
24. Yan, S., Xu, D., Zhang, B., Zhang, H.-J., Yang, Q., Lin, S.: Graph embedding and extensions: a general framework for dimensionality reduction. IEEE Trans. Pattern Anal. Mach. Intell. **29**(1), 40–51 (2007)
25. Yang, C., Liu, Z., Zhao, D., Sun, M., Chang, E.Y.: Network representation learning with rich text information. In: IJCAI, pp. 2111–2117 (2015)