Enhancing the Network Embedding Quality with Structural Similarity

Tianshu Lyu
Peking University
Department of Machine Intelligence
lyutianshu@pku.edu.cn

Yuan Zhang
Peking University
Department of Machine Intelligence
yuan.z@pku.edu.cn

Yan Zhang
Peking University
Department of Machine Intelligence
zhy@cis.pku.edu.cn

ABSTRACT

Neural network techniques are widely used in network embedding, boosting the result of node classification, link prediction, visualization and other tasks in both aspects of efficiency and quality. All the state of art algorithms put effort on the neighborhood information and try to make full use of it. However, it is hard to recognize core periphery structures simply based on neighborhood.

In this paper, we first discuss the influence brought by random-walk based sampling strategies to the embedding results. Theoretical and experimental evidences show that random-walk based sampling strategies fail to fully capture structural equivalence. We present a new method, SNS, that performs network embeddings using structural information (namely graphlets) to enhance its quality. SNS effectively utilizes both neighbor information and local-subgraphs similarity to learn node embeddings. This is the first framework that combines these two aspects as far as we know, positively merging two important areas in graph mining and machine learning. Moreover, we investigate what kinds of local-subgraph features matter the most on the node classification task, which enables us to further improve the embedding quality. Experiments show that our algorithm outperforms other unsupervised and semi-supervised neural network embedding algorithms on several real-world datasets.

CCS CONCEPTS

• **Computing methodologies** → *Neural networks*; *Learning latent representations*; • **Applied computing** → Sociology;

KEYWORDS

Network Embedding, Graphlet, Latent Representation

ACM Reference format:
Tianshu Lyu, Yuan Zhang, and Yan Zhang. 2017. Enhancing the Network Embedding Quality with Structural Similarity. In *Proceedings of CIKM'17 , Singapore, Singapore, November 6–10, 2017,* 10 pages.
https://doi.org/10.1145/3132847.3132900

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CIKM'17 , November 6–10, 2017, Singapore, Singapore
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-4918-5/17/11…$15.00
https://doi.org/10.1145/3132847.3132900

1 INTRODUCTION

Network (or graph) is a group of interconnected nodes and contains a wealth of information on the relationships between every pair of nodes. The analysis of graph is required in almost every field, for instance, online social network [4], biological research [23], credit rating [9] and so on. Birds of a feather flock together and people always have certain characteristics in common with their friends surrounded by. Therefore, researchers naturally utilize the neighborhood information of one node to predict its category.

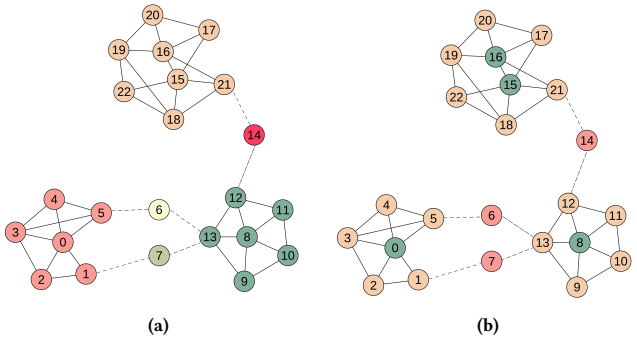
Unsupervised network embedding algorithms try to preserve the local relationships of each node in the graph. IsoMap [22], LLE [17] and Laplacian eigenmaps [1] are three classic dimensionality reduction and data representation algorithms. Finding the k nearest neighbors is the key step of these three algorithms. Classic algorithms can hardly tackle real networks due to the huge computational complexity of eigen-decomposition.

DeepWalk [15] first introduces deep learning techniques word2vec to the network embedding task. The authors pioneered the similarity between random walks and natural language. Later, more work [2, 7, 19, 28] is presented to improve DeepWalk by extending the definition of neighborhood and capturing neighborhood information from different levels of scope, namely first-order proximity, second-order proximity and higher-order proximity. Using random walks to capture the local structure is truly a neat idea, which makes it possible to build representations of big networks.

Although neighbors have been proved to be very significant features in all the state-of-art network embedding algorithms, *Structural Similarity* plays an irreplaceable role in various tasks ranging from node classification to visualization. Figure 1 shows two extreme cases when classifying nodes. Figure 1a is the classic way which is adopted by most algorithms. Nodes will be predicted to have same labels if they share many friends and connect with each other closely. The strategy focuses on neighbors. On the other hand, Figure 1b shows another strategy, that is dividing nodes by their structural similarity, namely *structural equivalence* defined in [7]. There are three groups altogether, core nodes, peripheral nodes and hub nodes under this criteria for network partition. These different starting points induce different definitions of similar nodes: (1) Densely connected nodes or (2) Nodes with similar network positions.

Most state-of-art algorithms, unfortunately, only concern about densely connected nodes. These two strategies are not in conflict but complementary, each with its own sphere of competence. For instance, if the node labels are about customers' interests, although the structure knowledge would contain invaluable information, blindly relying on it is not a good idea. Neighbors can provide much more reliable indications than structure similarity

RIGHTSLINK



GDV of node 0,2,8

		Orbit													
		1	2	3	4	5	6	7	8	9	10	11	12	13	14
Node ID	0	0	5	5	0	0	0	0	0	0	0	6	0	3	1
	8	0	5	5	0	0	0	0	0	0	0	5	0	5	0
	2	4	1	2	0	2	0	0	0	2	2	0	2	1	0

$E(0,8) = 6^{0.5}$, $E(0,2) = 98^{0.5}$

(c)

Figure 1: In (a) and (b), if two nodes are connected by dotted line, it means that there are a lot of intermediate nodes between them. The color of the node indicates its group. (a) and (b) present two cases that classifying nodes from two different fundamental angles. In (a), tightly connected nodes are in the same group. In (b), nodes with similar structural position are grouped into one partition. (c) is the Graphlet Degree Vector (GDV) of node 0,2,8 in (b). We only show 14 dimensions of each node. $E(\cdot,\cdot)$ is the Euclidean distance between two nodes.

does. If the labels are about the social status, topological information does matter. The most sensible approach is to make a balance between these two strategies.

Structural equivalence is being discussed not only in social role mining task [3], but in recent network embedding algorithms as well [7]. We support the motivation that both kinds of similar nodes should be taken into consideration. However, we argue that **embedding algorithms simply relied on random-walk sampling are not capable of capturing *Structural equivalence* effectively.** In this paper, we propose a framework **SNS** considering both **S**tructural and **N**eighborhood **S**imilarity. It can be used to refine all the word2vec-based network embedding algorithms.

We conclude three main contributions of our research:

(1) We investigate the relation between random-walk based sampling strategies and network embedding results theoretically and experimentally and point out the weakness (overestimation) of random-walk based sampling.
(2) We propose a general and robust neural network framework that can effectively utilize both neighbor information and local-subgraph similarity to learn node embeddings. This is the first framework that combines these two aspects as far as we know.
(3) We conduct some experimental studies to gain insight about how to make a rational use of structural information.

2 RELATED WORK AND BACKGROUND

2.1 Network Embedding

Networks contains rich information of which, however, we cannot make full use easily and quickly. Algorithms on graphs, even those as simple as calculating distances between any pairs of nodes, are usually costly. Representation learning can help to extract useful information and transform network data into an easy-used one. The obtained network embeddings can be used as input features in many downstream tasks such as classification and link prediction, which obviates the need for complicated and time-consuming methods directly applied on graphs. We refer the reader to [6] for more comprehensive details.

Traditional methods [20, 21] mainly focus on dimension reduction. By the techniques of matrix factorization, traditional methods project the adjacency matrix to a low-dimension space. As this kind of algorithms are first designed for general high-dimension data, they do not fit the typical network data. Networks are sparse at most time and node degree follows power-law distribution. The optimization from a global view cannot provide a satisfactory result for classification and other downstream applications. Just as discussed in Section 1, local information, namely neighborhood, is much more significant than global information.

Neural network embedding algorithms attract many attentions as the popular model, word2vec, in natural language processing seems to perfectly fit this task and achieves much better effects than classic methods do. The main reason is that the word frequency follows a power law distribution, just as the degree distribution in a graph. Besides *homophily* is pervasive in both languages and graphs, which means that both words and nodes can be predicted by their surrounding words and nodes. The difference between the state-of-art neural network based algorithms lies in the neighborhood sampling strategies. DeepWalk [15] uses depth-first search in order to sample the neighborhood of the target node. The depth is set to 2 by default. GraRep [2] also uses DFS but the depth is larger. LINE [19] uses breadth-first search as well as depth-first search. The number of step are all constrained below two. node2vec [7] also uses both two kinds of search methods and finds the balance point by semi-supervised learning. It selects the optimal balance point with grid search method, which is time-consuming. Furthermore, sampling strategy based on biased random walks is slower than the unbiased one. Obviously, none of these algorithms take structural similarity into consideration. SDNE [24] is not based on word2vec framework but deep autoencoder instead. It has the same goal as LINE does, considering both first-order proximity and second-order proximity.

GraRep [2] and node2vec [7] discuss local structure and structural equivalence. However, it is questionable whether the structural similarity is actually used during the learning process. Instead, they control the sampling process and capture neighborhood information from different levels of scope. Details will be discussed in the following sections. Our framework is the first to directly leverage local-subgraph information to learn network embedding.

RIGHTSLINK

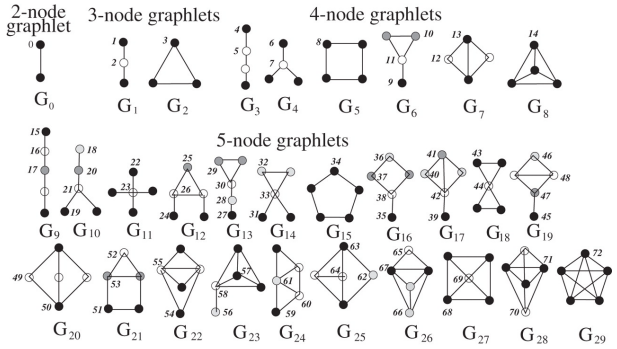


Figure 2: Graphlets with 2-5 nodes and automorphism orbits [8]. Nodes of the same color belong to the same orbit within that graphlet.

2.2 Structural Similarity

Computing similarities between structured objects (network) is a hot topic in recent years. Methods are mainly based on graphlets, subtree patterns and random walks [25]. In the network embedding task, we focus on nodes rather than the whole network structure, which means that we need a metric describing the structural characteristics of a single node. This topic has been given much consideration in the real fields such as biology and social science [26]. We first introduce graphlets and graphlet-based network distance measures.

Graphlets are small, connected, non-isomorphic, induced subgraphs of a big graph. As shown in Figure 2, there are altogether 30 graphlets with 2-5 nodes. Orbit indicates different node position of the graphlets. Symmetrical nodes have the same orbit number and there are 73 orbits of 30 graphlets. The *Graphlet Degree Vector* (GDV) of a node generalizes the notion of a node's degree into a 73-dimensional vector, of which each components represents the number of times node n is touched by a graphlet at orbit i (where $i \in [1, 73]$). If i equals 0, the number equals its degree. The structural similarity between nodes is denoted as the Euclidean distance of their GDV. In Figure 1c, we calculate the GDV of node 0,2,8 in Figure 1b for example. We only show 14 dimensions of each node, corresponding to two 3-node graphlets and six 4-node graphlets. $E(\cdot, \cdot)$ denotes the Euclidean distance between two nodes' GDV, which can describe the structural similarity clearly. Besides Euclidean distance, GDV distance can also be defined in other forms, for instance, Spearman correlation.

There are many graphlet counting algorithms that can provide precise results on small graph and approximate results on big graph with a quick speed. In this paper, we use *orca* [8] to help us calculate GDV of each node. The code is available on the authors' website.

3 RANDOM-WALK BASED SAMPLING

Network embedding algorithms based on random walk preserve higher-order proximity between nodes by maximizing the probability of occurrence of subsequent nodes in fixed length random walks. In this section, we take a deeper look at the sampling strategy based on random walks and the proximity captured by random walks.

Authors of [15] give an analogy for a word and its context in natural language to the target node along with its neighbors in a graph. The learning process leverages the co-occurrence probability of the nodes that appear within a window in a random walk. Node pairs with high co-occurrence probability are regarded as neighbors. As the size of window is usually no less than two, we call this kind of neighbors as higher-order proximity. We denote the probability that node i and j are successively visited at an interval of r steps as $P(i, j, r)$.

Consider an undirected graph G with N nodes and m edges. The adjacency matrix $\mathbf{A}$ is symmetric and the entries equal 1 if there is an edge between two nodes and 0 otherwise. Vector $\mathbf{d} = \mathbf{A1}$, where $\mathbf{1}$ is a $N \times 1$ vector of ones and each entry of $\mathbf{d}$ is the node degree. Graph G has an associated random walk in which the probability of leaving a node is split uniformly among the edges.

$$\mathbf{p_{t+1}} = \mathbf{p_t D^{-1} A} \equiv \mathbf{p_t M}, \tag{1}$$

where $\mathbf{p}$ is the probability vector and $\mathbf{D}$ is the diagonal matrix of degree: $\mathbf{D} = diag(\mathbf{d})$. $\mathbf{M}$ is the transition matrix. Given by $\pi = \pi\mathbf{M}$, the stationary distribution of this Markov chain is $\pi = \mathbf{d}^\top/2m$, $P(i) = d_i/2m$.

For a walk starting at node i, the probability that we find it at j after r steps is given by

$$P(j, r|i) = [\mathbf{M^r}]_{\mathbf{ij}},$$

$$P(i, j, r) = P(j, r|i)P(i) = \frac{d_i}{2m}[\mathbf{M^r}]_{\mathbf{ij}} \propto [\mathbf{AM^{r-1}}]_{\mathbf{ij}}. \tag{2}$$

Equation 2 demonstrates that the co-occurrence probability of two arbitrary nodes i and j is mainly decided by the degree of the nodes in the path between i and j and has nothing to do with the degree of i or j. Note that the path length is restricted to the window size. Apparently, among all the intermediate nodes in the feasible path between i and j, nodes with smaller degree and in shorter path contribute more to the co-occurrence probability. To put it more generally, two nodes are regarded as neighbors if the connection between them is *strong* and *direct*. As shown in Figure 3, the more short feasible paths between two nodes (*strong*) and the lower degree the intermediate nodes are (*direct*), the higher the co-occurrence probability is.

In the learning process, nodes with similar neighborhood will have similar latent representations. The scope of neighborhood is decided by the size of window. Two nodes in the network separated by distance longer than the window size have no chance to calculate the mutual similarity. In other words, random-walk based sampling strategy only captures the higher-order proximity within the neighborhood of the target node. Besides, the higher-order proximity of two nodes is mainly determined by how strong and direct the connections are. It is not capable of capturing the structural equivalence over the whole graph.

Grover et. al. propose a more general method, node2vec [7], which simulates biased random walks by introducing two parameters p and q and controlling the search procedure interpolating between BFS and DFS. These two parameters actually modify the transition matrix $\mathbf{M}$. As stated in the last paragraph, smaller return parameter p encourages stronger connections between neighbors. Smaller in-out parameter q, on the other hand, encourages more

RIGHTSLINK

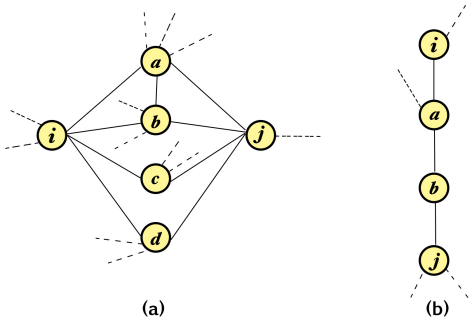


Figure 3: Using random-walk based sampling strategy, two nodes, i and j, are more likely neighbors if the connections between them are (a) strong: a number of feasible paths and (b) direct: the intermediate nodes have few connections. Nodes in this figure also connect with other nodes that are not shown in this figure. These kind of edges are denoted by dotted lines.

direct connections. This work defines a flexible notion of neighborhoods and we argue that (1) the strength and directness of connections should not be seen as structural equivalence and (2) any topological information outside the window cannot be captured.

In Section 6.1, we visualize the embedding results provided by DeepWalk, node2vec and our proposed algorithm, demonstrating our point of view experimentally.

4 PRE-PROCESSING STEP: TO OBTAIN THE STRUCTURALLY SIMILAR NODES

Now that random-walk based sampling strategies cannot capture structure equivalence, we turn to other graph mining techniques for additional capabilities. In this section, we discuss the details that how we obtain the local-subgraph information in the pre-processing step and use it in the network embedding task.

Peripheral orbits are informative. In Section 2.2, we have presented the use of graphlets and orbits. The number of orbits grows quickly as the size of graphlets increases. Too many features, however, will result in over-fitting and not all orbits contribute equally to a certain task. Therefore, we evaluate the importance of different orbits on the node classification task [5] with the help of Random Forest [14].

Random Forest consists of a number of decision trees. In the decision trees, every node is a condition on a single feature, designed to split the dataset into two parts so that similar response values end up in the same set. Impurity is to measure the homogeneity of the target variable within the subsets. For classification, it is typically either Gini impurity or information gain. Thus when training a tree, it can be computed how much each feature decreases the weighted impurity in a tree. For a forest, the impurity decreased from each feature can be averaged and we call this measure feature importance. Note that features with more categories and with less correlated features are considered to be more important when using the impurity based ranking.

We use random forest implemented by sklearn [14] and calculate the orbit importance I_o as stated before. The experiments are repeated on several datasets. Table 1 shows the relative importance of each orbits RI_o on BlogCatalog. We can get similar observations on the other datasets.

Table 1: The relative importance of the orbits RI_o of the node classification task on BlogCatalog

orbit	*RI%*	orbit	*RI%*	orbit	*RI%*	orbit	*RI%*	orbit	*RI%*
0	0.05	1	21.18	2	1.79	3	0.14	4	84.58
5	62.26	6	95.38	7	10.94	8	21.27	9	81.94
10	63.48	11	8.50	12	41.98	13	6.15	14	3.71
15	92.27	16	76.27	17	85.71	18	86.70	19	93.90
20	82.76	21	62.11	22	86.08	23	20.55	24	86.28
25	83.70	26	65.51	27	88.68	28	81.02	29	82.77
30	63.89	31	100.00	32	76.21	33	20.14	34	75.33
35	95.19	36	77.29	37	82.30	38	44.18	39	94.31
40	77.64	41	65.50	42	18.75	43	80.58	44	11.52
45	87.21	46	80.75	47	58.46	48	65.70	49	83.55
50	24.42	51	76.65	52	80.50	53	43.72	54	85.05
55	15.38	56	91.93	57	66.20	58	14.91	59	77.65
60	55.84	61	13.73	62	74.85	63	31.37	64	41.12
65	83.61	66	59.60	67	16.18	68	39.53	69	7.20
70	49.11	71	14.99	72	12.14				

$$RI_o = \frac{I_o}{\max\limits_{0 \leqslant i \leqslant 72} (I_i)} \times 100\%$$

We observe that orbit 0 (node degree) is of little importance from the perspective of node classification. A more general idea is that peripheral orbit (orbits with less degrees) is more important than the core orbit (orbits with more degrees) for any graphlet. For instance, orbit 1 is more important than orbit 2 in G_1. In G_{12}, orbit 24 has the greatest importance while orbit 26 has the smallest one. This can be verified by the Table 1 and Figure 2.

One reason of this interesting phenomenon might be the dependency between graphlet degrees. Take any node with more than two neighbors as an example, it can form either G_1 or G_2 with any two of its neighbors. If we denote C_i as the graphlet degree of orbit i, we then have

$$\binom{C_0}{2} = C_2 + C_3.$$

Core orbits are consequently more likely to be redundant, as it might be expressed by the sum of some peripheral orbits.

Because of the analysis above, peripheral orbits should weigh more than core orbits when we calculate the similarity of two GDVs.

Limit the scope of similar nodes. In order to cooperate the network embedding task, we try to leverage local-subgraph similarity information. For a target node, its local neighbors and structurally similar nodes are involved in the learning process. Structurally similar nodes can be chosen from the whole network or only the neighborhood of the target node. The scope of structurally similar nodes to be considered depends on the real task. In certain circumstance, nodes connected together in the network are more likely to have the same labels than those without connections between them. The maximum step between the similar node and the target node is supposed to be small. However, there are also cases where structural similarity is the major factor and the maximum step has to be bigger.

RIGHTSLINK

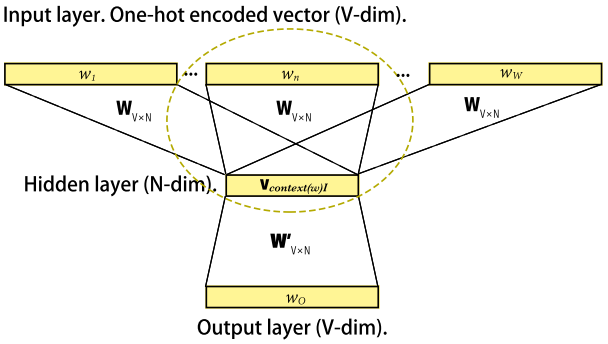


Figure 4: The framework of CBOW. $w_1, w_2, \cdots, w_W$ are the context words of the target word. The vocabulary size is V. The window size is W. W is the weight matrix between the input layer and the hidden layer. Each row of W is the N-dim word vector $\mathbf{v}_w$. And similarly, $\mathbf{v'}_w$ is from $\mathbf{W'}$.

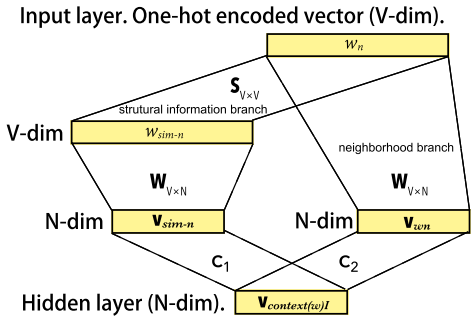


Figure 5: Combine the structural information with the network embedding. Given space limitations, we only demonstrate the complete neural network architecture of w_n. For the other $W-1$ context words, they also have new architectures as w_n does.

In the pre-processing step, graphlet counting algorithms first produce Graphlet Distribution Vector for every node. Next for each node, find its K nearest neighbors in the GDV space, i.e. the K neighboring nodes which have the smallest cosine distance from it. Note that searching structurally similar nodes has to be restricted in the higher order S^{th} degree neighborhood of the target node $\mathcal{N}^S_{v_i}$ as:

$$\mathcal{N}^S_{v_i} = \{v_j \mid \mathbf{A}_{ij} = 1 \vee \mathbf{A}^2_{ij} \geqslant 1 \vee \cdots \vee \mathbf{A}^S_{ij} \geqslant 1\}.$$

This is the set of all nodes at a distance no more than S from target node v_i. Search results are preserved in a sparse matrix S, where s_{ij} equals the similarity score if j is one of the K nearest neighbors of i and 0 otherwise ($s_{ij} \in [0, 1)$ and $\sum_j s_{ij} = 1$). There are altogether K non-zero numbers in each row of the matrix S. To sum up, the pre-processing is controlled flexibly from the following five aspects:

(i) The metric to evaluate the similarity of GDV.
(ii) O: The number of orbits to be considered.
(iii) R: The weight of different orbits when calculating the similarity of GDV.
(iv) K: The number of the most similar nodes.
(v) S: The maximum step between the similar node and the target node.

5 NETWORK EMBEDDING POWERED BY STRUCTURAL SIMILARITY

We propose a neural network architecture that leverages both local connected neighbors and topological information to learn network embedding.

5.1 CBOW Model with Negative Sampling

Although our framework can be used to refine all the word2vec-based network embedding algorithms, we use CBOW [12] as the basis of our framework due to the space limitations. Comparing with skip-gram model, CBOW model performs better on dataset with short sentences but high number of sample sets (larger dataset).

In the field of NLP, similar words have similar contexts and CBOW aims to predict a word by its context. The loss function is as follows. C is the vocabulary set and w is the word to be predicted.

$$\mathcal{L} = \sum_{w \in C} \log p(w|Context(w)).$$

CBOW takes the one-hot encoder of the context words w_I as input and the average of the context word vectors as hidden layer, where W is the number of context words of word w and $\mathbf{v}_{w_i}$ is the vector of word w_i.

$$\mathbf{h} = \mathbf{v}_{w_I} = \frac{1}{W}(\mathbf{v}_{w_1} + \mathbf{v}_{w_2} + \ldots + \mathbf{v}_{w_W}).$$

w_O is the output of CBOW and y_j is the j-th unit of the output layer. Note that $\mathbf{v}_w$ and $\mathbf{v'}_w$ are the input vector and output vector of the word w respectively.

$$p(w|Context(w)) = y_j = \frac{\exp(\mathbf{v'}^T_{w_j}\mathbf{v}_{w_I})}{\sum_{j'=1}^{V}(\mathbf{v'}^T_{w_j}\mathbf{v}_{w_I})}.$$

The *softmax* function is very difficult to optimize because of the huge amount of calculation in the denominator. One of the effective solutions is negative sampling, which approximates the *softmax* function by performing logistic regression to k noise samples. As shown in [12], the probabilistic distribution for the negative sampling process is a unigram distribution raised to the $\frac{3}{4}th$ power. We denote $P_n(w)$ as the noise distribution and σ is the logistic function. $\mathcal{W}_{neg}$ is the set of k samples sampled based on $P_n(w)$.

$$P_n(w) = \frac{U(w)^{\frac{3}{4}}}{Z},$$

$$\mathcal{L} = \log \sigma(\mathbf{v'}^T_{w_O}\mathbf{v}_{w_I}) + \sum_{w_j \in \mathcal{W}_{neg}} \log \sigma(-\mathbf{v'}^T_{w_j}\mathbf{v}_{w_I}).$$

5.2 Enhanced by the Structural Information Branch

We propose a new framework in order to boost the learning process with the structural similarity information. Figure 4 presents the framework of CBOW. We take the dotted portion in Figure 4 as an example and its corresponding new architecture is showed in

RIGHTSLINK

Figure 5. In this model, the target node is predicted by its neighborhood (contextual information, referred to as neighborhood branch), as well as the structurally similar nodes nearby (referred to as structural information branch). As discussed in the last section, the sparse similarity score matrix $\mathbf{S}$ is derived from the pre-processing step. Each row of $\mathbf{S}$ contains K non-zero items and s_{ij} is referred to as the similarity score of node i and node j. Neighborhood branch and structural information branch share one embedding matrix $\mathbf{W}$. Each row of $\mathbf{W}$ represents the input embedding vector v of a node. The corresponding vector of structural information is written as

$$\mathbf{v}_{sim\text{-}n} = \sum_{m=1}^{V} s_{nm}\mathbf{v}_{w_m}.$$

The similarity score s_{nm} is regarded as the weight assigned to the vector of w_m, one of the top-K topologically similar nodes of w_n.

An aggregated representation of the context of the target input node is

$$\mathbf{v}_{w_I} = \frac{1}{W}\sum_{n=1}^{W}(c_1(deg(w_n))\mathbf{v}_{w_n} + c_2(deg(w_n))\mathbf{v}_{sim\text{-}n}),$$

where c_1 and c_2 are the parameters regulating the proportion of the two kinds of information ($c_1, c_2 \in (0, 1)$). Usually, the neighborhood of a high degree node can provide enough information for label prediction. The node with few edges, however, depends much on structural information and less on its neighborhood. Thereby, the value of c_1 is larger than c_2 for high degree nodes and vise versa. Nodes are sorted in descending order of node degree and divided into C levels. Nodes at the same level share the same c_1 and c_2.

In conclusion, the model takes the weighted average of the node vectors of both the target node's neighbors and their similar nodes as input.

5.3 Learning Process

Matrix $\mathbf{S}, \mathbf{W}, \mathbf{W}'$ and the value of c_1, c_2 are updated in the back-propagation process. The partial derivative of $\mathcal{L}$ with regard to the net input of the output unit w_j is as follows:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}'^{T}_{w_j}\mathbf{v}_{w_I}} = \begin{cases} \sigma(\mathbf{v}'^{T}_{w_j}\mathbf{v}_{w_I}) - 1, & \text{if } w_j = w_O; \\ \sigma(\mathbf{v}'^{T}_{w_j}\mathbf{v}_{w_I}), & \text{if } w_j \in \mathcal{W}_{neg}. \end{cases}$$
$$= \sigma(\mathbf{v}'^{T}_{w_O}\mathbf{v}_{w_I}) - t_j.$$

where t_j is the indicator. Using the chain rule, we further get:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}'_{w_j}} = \frac{\partial \mathcal{L}}{\partial \mathbf{v}'^{T}_{w_j}\mathbf{v}_{w_I}} \cdot \frac{\partial \mathbf{v}'^{T}_{w_j}\mathbf{v}_{w_I}}{\partial \mathbf{v}'_{w_j}} = (\sigma(\mathbf{v}'^{T}_{w_O}\mathbf{v}_{w_I}) - t_j)\mathbf{v}_{w_I},$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}_{w_I}} = \sum_{w_j \in \{w_O\}\cup\mathcal{W}_{neg}} \frac{\partial \mathcal{L}}{\partial \mathbf{v}'^{T}_{w_j}\mathbf{v}_{w_I}} \cdot \frac{\partial \mathbf{v}'^{T}_{w_j}\mathbf{v}_{w_I}}{\partial \mathbf{v}_{w_I}}$$
$$= \sum_{w_j \in \{w_O\}\cup\mathcal{W}_{neg}} (\sigma(\mathbf{v}'^{T}_{w_O}\mathbf{v}_{w_I}) - t_j)\mathbf{v}'_{w_j} \equiv \Delta.$$

The corresponding update equation is as followed:

$$\mathbf{v}'^{(new)}_{w_j} = \mathbf{v}'^{(old)}_{w_j} - \eta(\sigma(\mathbf{v}'^{T}_{w_O}\mathbf{v}_{w_I}) - t_j)\mathbf{v}_{w_I},$$

$$c_1^{(new)} = c_1^{(old)} - \eta\mathbf{v}_{w_i}\Delta, \quad c_2^{(new)} = c_2^{(old)} - \eta\mathbf{v}_{sim\text{-}i}\Delta,$$

$$\mathbf{v}_{w_i}^{(new)} = \mathbf{v}_{w_i}^{(old)} - \eta c_1\Delta, \quad \mathbf{v}_{w_j}^{(new)} = \mathbf{v}_{w_j}^{(old)} - \eta c_2 s_{ij}\Delta,$$

$$s_{ij}^{(new)} = s_{ij}^{(old)} - \eta\mathbf{v}_{w_j}\Delta.$$

Note that matrix $\mathbf{W}$ is updated twice in one iteration, that is, one round by the nodes in the neighborhood and the other round by the topologically similar nodes. When the learning process is over, we take the matrix $\mathbf{W}$ as the learned node representations.

5.4 Variants

We briefly discuss some variants of our framework here. For different tasks and networks, we can choose proper variants.

Whose similar nodes are they? In our framework, we leverage the similar nodes of the context nodes rather than the target node. However, the latter plan is also feasible. The difference mainly lies in the scope of similar nodes to be chosen from. If we want to choose the nodes from a bigger scope (the maximum step away from the target node is 3 or bigger), the latter plan will lead to huge computational complexity of the pre-processing step. The current plan, on the contrary, achieves the goal by making the window-size $\mathcal{W}$ bigger. Larger window-size does not influence the efficiency significantly. Problem for the current plan is that the similarity is not transitive in some cases. Which plan is better might depend on the assortativity of the network.

Fixed value of c_1, c_2 and similarity matrix. In our framework, c_1, c_2, and $\mathbf{S}$ are updated in the learning process. Setting proper initial values and making them fixed can also achieve a good performance. Moreover, fixed parameters accelerate the learning process. The trick is setting bigger c_1 and smaller c_2 for high degree nodes and vise versa. The similarity scores seem to be less important. If node j is one of the top-K topologically similar nodes of node i, then we set $s_{ij} = 1/K$. Otherwise, $s_{ij} = 0$.

6 EXPERIMENTS

In this section we first visualize a small network using DeepWalk [15], node2vec [7] and our proposed algorithm separately, which demonstrates the arguments in Section 3 visually. Moreover, we show the performance of different network embedding algorithms on multi-label classification task. The parameter sensitivity will also be discussed, helping us have a deeper insight of the topological information and the algorithm efficiency.

6.1 Case Study: Visualization in 2-D space

Les Miserables co-appearance network consists of 77 nodes and 254 edges, where node corresponds to characters showed up in Les Miserables and edge corresponds to the co-appearance relationship. As the network is small, it is easy to visualize. Nodes in this network have various structural positions, making it easier to analyze different sampling strategies thoroughly. We compare our algorithm with DeepWalk [15] and node2vec [7]. Both of them are classic network embedding algorithm with random-walk based sampling strategy. Node2vec claims to be able to find structural equivalence when the parameters p and q are properly set. We use the same settings as they reported in the paper ($p = 1, q = 2$). DeepWalk is a special case of node2vec with $p = 1$ and $q = 1$. For simplicity,

RIGHTSLINK

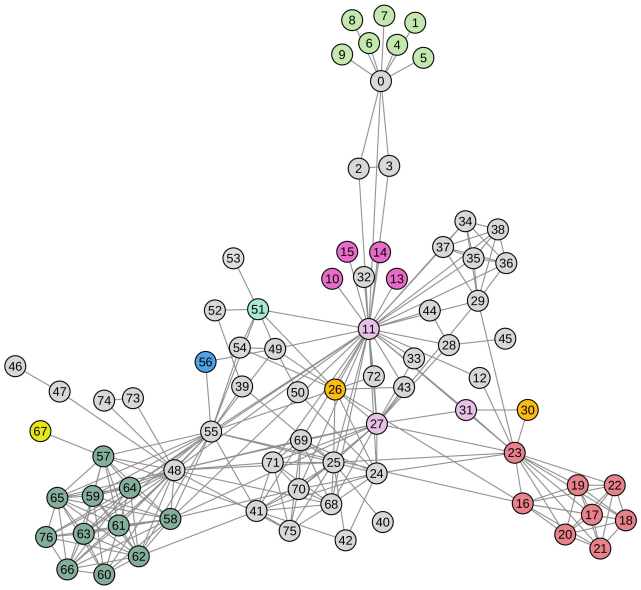


Figure 6: Visualization of Les Miserables co-appearance network using a force-directed layout: ForceAtlas2 [10].

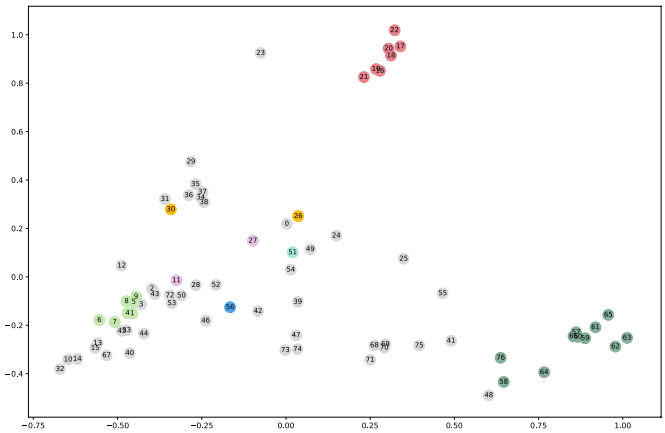


Figure 7: We use DeepWalk [15] to learn the latent representation of Les Miserables network ($p = 1, q = 1, d = 16$). Nodes are mapped to the 2-D space using the PCA package [13] with learned embeddings as input.

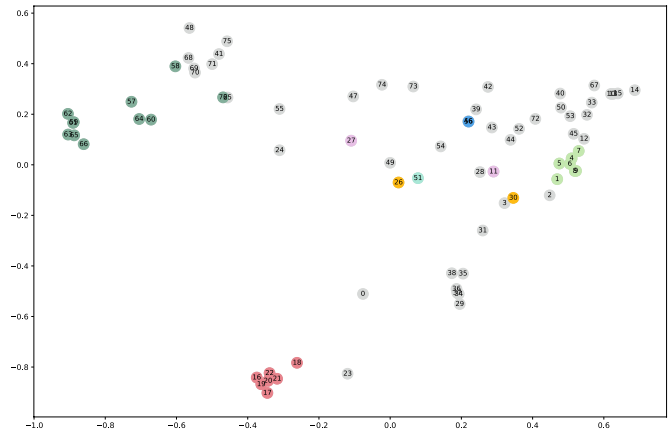


Figure 8: We use node2vec [7] to learn the latent representation of Les Miserables network ($p = 1, q = 2, d = 16$). Nodes are mapped to the 2-D space using the PCA package [13] with learned embeddings as input.

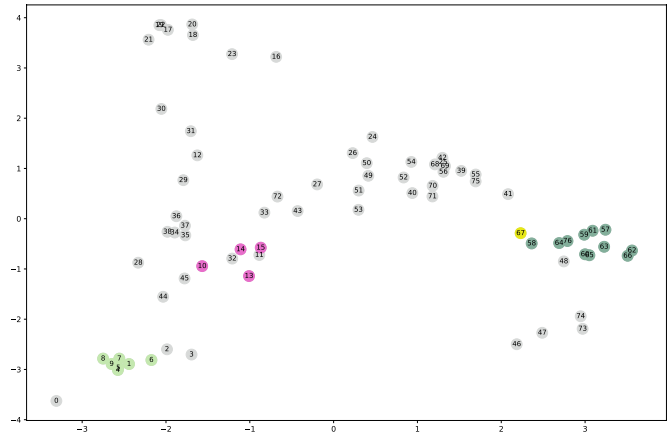


Figure 9: We use SNS to learn the latent representation of Les Miserables network. Nodes are mapped to the 2-D space using the PCA package [13] with learned embeddings as input.

the window size of these three algorithms are all set to be 4, which means that neighbors are chosen from the nodes at a distance no more than 2 from the target node.

Figure 6 is the visualization of Les Miserables co-appearance network using a force-directed layout: ForceAtlas2 [10]. Figure 7, Figure 8 and Figure 9 are the learned latent vectors provided by DeepWalk, node2vec and SNS respectively. Note that d, the dimension size of the latent space, is 16 and we use Principal Component Analysis to project the vectors to 2-dimensional space. Colors of nodes are uniform in Figure 6-9, making it easier for us to track them in different figures.

As stated in Section 3, random-walk based sampling strategies are sensitive to strong and direct connections. In Figure 6, both node 51 (light blue) and node 56 (dark blue) are in the neighborhood of node 26 (orange). There are many short paths between node 26 and 51, 26-51, 26-49-51, 26-54-51, 26-11-51 and etc. On the contrast, the only two short paths between node 26 and 56 are 26-49-56 and 26-55-56. Therefore, node 26 has stronger connection with node 51 and as shown in Figure 7, the distance between node 26 and node 51 is much shorter than it between node 26 and node 56. In Figure 6, node 11 (light purple), 16-22 (red) and 27 (light purple) are two-step neighbors of node 30 (orange). The intermediate node 23 has much

RIGHTSLINK

more connections than node 31 does. Therefore, in Figure 7, node 16-22 (red) are far away from node 30 (orange).

We also discuss the influence of biased random walk in Section 3 that smaller p encourages stronger connections between neighbors and smaller q encourages more direct connections. In our experiments, node2vec are set to $p = 1, q = 2$ and are therefore more sensitive to strong connections. Following the discussion of DeepWalk, the relationship between node 26 (orange) and 51 (light blue) is stronger compared with the one between node 26 and 56 (dark blue). In Figure 8, we can figure that node 26 and node 51 are quite close. The distance is much shorter than it is in Figure 7.

In Figure 6, node 1, 4-9 (light green) are structurally similar as they all have 1 edge connected to node 0. Node 57-65, 76 (dark green) are also a group of similar nodes as they all connect to each other and thus form a clique. In Figure 7 and Figure 8, we find that the locations of these two groups of nodes are quite different. Node 57-65, 76 (dark green) are separated from the other nodes, while node 1, 4-9 (light green) can hardly separate from the surrounding nodes. This difference proves that whether the random-walk based sampling strategy is biased or not, it is not capable of finding out the nodes with structural equivalence under any circumstances. The reason why node 57-65, 76 are away from other nodes is that the connections between them are much stronger and more direct than the rest. Other nodes can hardly be visited in the random walks started from node 57-65 and 67. Node 1, 4-9, on the contrary, do not have such local structure and the structural similarity of this group of nodes can not be captured by random-walk based sampling algorithms.

In Figure 9, these two groups of nodes are all separated from the rest of nodes. This improvement attributes to our consideration of graphlets distribution. Note that although nodes 10, 13-15 (dark purple) and etc. also only have one edge as node 1, 4-9 do, their local structural position are quite different in our opinion. For example, node 67 (yellow) connects to node 57 in Figure 6 and node 57-65 and 76 form a clique as we stated before. We consider node 67 is more similar with node 57-65 and 76. Node 13-15 all have one edge connected to node 11, which is the 2-step neighbor of node 1, 4-9. We consider node 10, 13-15 are similar with node 1, 4-9. These kinds of relationships are all reflected in the latent space as shown in Figure 9.

Through all the demonstrations above, we try to prove that random-walk based sampling strategies are not good at mining structural equivalence. The properties of random walks constrain the capability of this kind of algorithms. They only capture the strength and directness of the relationships between nodes, rather than the exact topological information of every node. SNS, with the help of graphlet distribution vector, is capable to mine structural equivalence.

6.2 Datasets and Baselines

In the following experiments, we choose three datasets that accord our starting-point. They all exist a mix of homophily and structural equivalences [7].

BlogCatalog [27] is the social blog directory which manages the bloggers and their blogs. The network depicts the contact between users and the user labels represent their interests. Users tend to

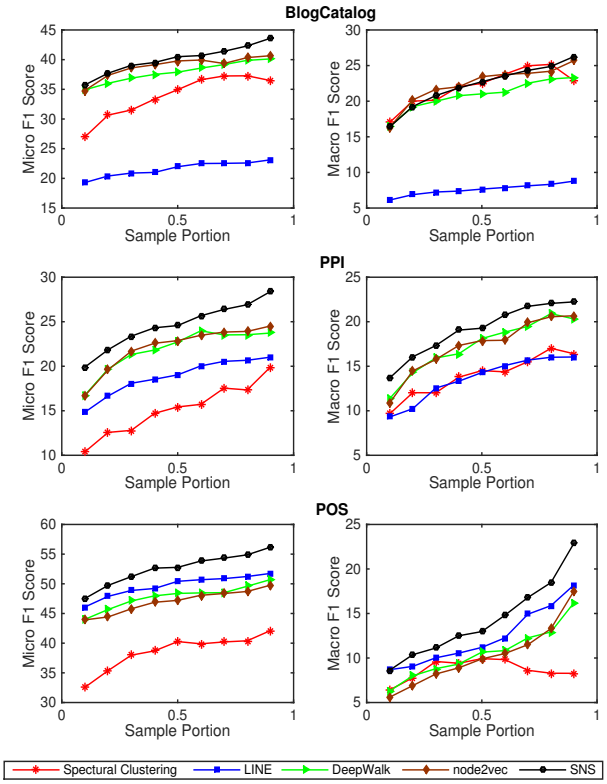


Figure 10: The Macro-F1 Score and Micro-F1 Score of different algorithms on varying the sampling portion used for training.

have same hobbies as their close friends. Strangers but with similar structural positions are also possible. The network contains 10,312 nodes, 333,983 edges and 39 labels.

Protein-Protein Interactions [18] is a subgraph of the PPI network for Homo Sapiens. Labels stand for the protein biological states. The network contains 3,890 nodes, 76,584 edges and 50 labels.

POS [11] is a co-occurrence network of words appearing in the first million bytes of the Wikipedia dump. A part of speech (POS) is a category of words which have similar grammatical properties. Words that always show together have a high probability to be similar meaning. And words with similar network structure also tend to be with the same part of speech. The network contains 4,777 nodes, 184,812 edges and 40 labels.

We choose a classic method aiming at dimension reduction and three representative neural network embedding algorithms as baselines.

Spectral Clustering [21] is based on matrix factorization and aims at minimizing Normalized Cut. We set $d = 500$, the same setting in [21].

DeepWalk [15] is the first embedding algorithm that brings the deep learning technology. It is an unsupervised learning algorithm. It captures the neighborhood information of each node by random

RIGHTSLINK

walks and uses skip-gram model with negative sampling. We set $d = 128$, $r = 10$, $l = 80$, $k = 10$, the same setting in [15].

LINE [19] defines a loss function based on 1-hop and 2-hop relational information. It learns $d/2$ dimensions of the node vector by these two parts of information respectively and combines them directly as the final output. In a supervised learning task, it finds the weighting of dimensions based on training data and achieves a better performance. In our experiments, we try the unsupervised mode of LINE [19] and set $d = 128$, $r = 10$, $l = 80$, $k = 10$, the same setting in [19].

node2vec [7] simulates biased random walks over the underlying network. It uses parameter p and q to balance the BFS strategy with DFS strategy.DeepWalk is a special case of node2vec where $p = q = 1$. node2vec is a semi-supervised algorithm and it need 10% labeled data of the network to decide the value of p and q. We use the value of p and q as showed in the authors' paper [7].

The settings of our algorithm SNS is as follows: in the pre-processing step, $K = 5$, $S = 1$, $O = 14$, $R = 9$. In the learning step, we use the same parameters as DeepWalk does and $C = 5$. In the pre-processing step, we use *orca* [8], a very efficient algorithm for graphlet enumeration.

As reported in the paper [8], on a desktop computer (Intel Core 2, 2.67 GHz), *orca* only takes 2.5s to count the four-node graphlets of a network with 25,368 nodes and 75,004 edges. SNS spends more time than DeepWalk does, as more parameters are involved in the learning process. The sampling step of node2vec, similarly, is very time-consuming, comparing with the normal random-walk sampling used by DeepWalk and SNS.

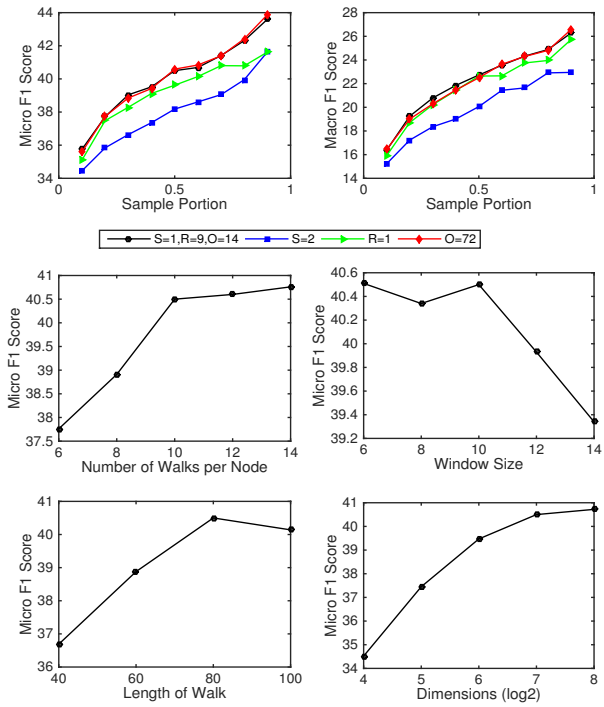


Figure 11: Parameter Sensitivity of SNS on the BlogCatalog network.

6.3 Multi-label Classification

This task uses the exact same datasets and experimental procedure as presented in [7]. The node vectors are the input to a one-vs-rest logistic regression implemented by sklearn [14]. We sample a portion of the labeled nodes as training data and use the rest nodes as test data. This process is repeated 10 times, and we show the average Micro-F1 Score and Macro-F1 Score in Figure 10.

We can easily figure out that SNS has advantages over other algorithms on these three datasets. Spectural Clustering is a competitive algorithm only on BlogCatalog dataset. DeepWalk seems to be the most stable algorithm among the baselines. This performance suggests that different neighborhood sampling strategies are not so reliable in different fields. Designing special sampling strategies will result in limited application scope. In BlogCatalog dataset, the advantage of SNS is not as evident as it in the other two datasets. Neural network based algorithms have close scores. The reason might be that node labels of BlogCatalog datasets depend more on the node neighborhood. In PPI dataset and POS dataset, the obvious advantages have to be attributed to the help of structural similarity information.

Note that node2vec has to utilize supervised data to decide the best parameter, while SNS is unsupervised and achieves better performance. This illustrates that our intuition, combining the structural similarity with neighborhood information, is an important knowledge of multi-label classification task.

6.4 Parameter Sensitivity

We test our algorithm with different settings on BlogCatalog dataset. The results are similar for the other datasets. The first two sub-figures of Figure 11 focus on the settings of structural similarity. S stands for the maximum step between the similar node and the target node. R stands for the weighting factors between peripheral orbits and core orbits when calculating the GDV distance. O stands for the number of orbits to be considered (the dimension of GDV). If the node labels is relevant to the node distance in the network, bigger S will bring many nodes far from the target node and have a negative effect on the label prediction task. As discussed in Section 4, peripheral orbits are more informative. Thereby, they weigh more than core orbits. The number of orbits does not matter much. Using more orbits to calculate the topological similarity between nodes does not contribute to a better result. Fourteen orbits from graphlets with 2-4 nodes is enough.

The rest sub-figures are about the parameters of random walks. Window size k has little effect on the results. We attribute the good performance of small window size to the help of topological information. A big window size brings much noisy information and makes the embedding quality worse. Number of walks per node r and walk length l have relatively large impact. These two parameters control the scale of training data. Increasing dimension d improves the result. When d is bigger than 100, the improvement is less obvious.

RIGHTSLINK

6.5 Discussion: Local Versus Global

The proposed model SNS is capable of capturing both local and global structural characteristics. We are going to discuss about their feasibility and practicality. Utilizing global information of the network is a difficult field and we have to acknowledge that the computational cost of SNS is huge when S is a large value. Some related work [2, 16] suffers from the same drawback. A possible way to improve the performance might be designing heuristics and limiting the search scope. On the other hand, it seems that in most real tasks global information is less useful than local information. Long distance, in the network indicates the potential gaps of node features. Structural similarity is less discriminative than spatial proximity to certain extent. In conclusion, local structural information is easy to be captured and more commonly used in real tasks. Meanwhile, we are looking forward to the new network embedding algorithms effectively capturing global structural information.

7 CONCLUSION AND FUTURE WORK

In this paper, we analyze the characteristics and the downsides of random-walk based sampling strategies and propose a network embedding framework combining both structural similarity and neighborhood information. We capture the structural information by graphlet degree vector and make full use of it in our learning framework. This method alters the neural networks used by the word2vec algorithms to contain a topological information factor, introducing new layers with several tunable parameters. SNS is the first framework that leverages network structure directly and makes it possible no matter in local area or in a global scope of the network. The experiments show that structural similarity does help in the node classification task.

Tasks, including node classification, link prediction and visualization, are all relevant with local network structure. If there exists any task that has to use global information of the network, algorithms based on random walks are not suitable any more. In conclusion, future work includes finding new application scenarios and new framework supporting capturing global network information.

8 ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments and helpful suggestions. This work is supported by 973 Program under Grant No.2014CB340405, NSFC under Grant No.61532001 and No.61370054, and MOE-RCOE under Grant No.2016ZD201.

REFERENCES

[1] Mikhail Belkin and Partha Niyogi. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation* 15, 6 (2003), 1373–1396.
[2] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 891–900.
[3] Sarvenaz Choobdar, Pedro Ribeiro, Srinivasan Parthasarathy, and Fernando Silva. 2015. Dynamic inference of social roles in information cascades. *Data Mining and Knowledge Discovery* 29, 5 (2015), 1152–1177.
[4] Linton C Freeman. 2000. Visualizing social networks. *Journal of social structure* 1, 1 (2000), 4.
[5] Robin Genuer, Jean-Michel Poggi, and Christine Tuleau-Malot. 2010. Variable selection using random forests. *Pattern Recognition Letters* 31, 14 (2010), 2225–2236.
[6] P. Goyal and E. Ferrara. 2017. Graph Embedding Techniques, Applications, and Performance: A Survey. *ArXiv e-prints* (May 2017). arXiv:1705.02801
[7] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 855–864.
[8] Tomaž Hočevar and Janez Demšar. 2014. A combinatorial approach to graphlet counting. *Bioinformatics* 30, 4 (2014), 559–565.
[9] Shian-Chang Huang. 2009. Integrating nonlinear graph based dimensionality reduction schemes with SVMs for credit rating forecasting. *Expert Systems with Applications* 36, 4 (2009), 7515–7518.
[10] Mathieu Jacomy, Tommaso Venturini, Sebastien Heymann, and Mathieu Bastian. 2014. ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software. *PloS one* 9, 6 (2014), e98679.
[11] Matt Mahoney. 2009. Large text compression benchmark. *URL: http://www. mattmahoney. net/text/text. html* (2009).
[12] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
[13] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12, Oct (2011), 2825–2830.
[14] 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. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[15] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.
[16] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning Node Representations from Structural Identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 385–394.
[17] Sam T Roweis and Lawrence K Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science* 290, 5500 (2000), 2323–2326.
[18] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. 2006. BioGRID: a general repository for interaction datasets. *Nucleic acids research* 34, suppl 1 (2006), D535–D539.
[19] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 1067–1077.
[20] Lei Tang and Huan Liu. 2009. Scalable learning of collective behavior based on sparse social dimensions. In *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 1107–1116.
[21] Lei Tang and Huan Liu. 2011. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery* 23, 3 (2011), 447–478.
[22] Joshua B Tenenbaum, Vin De Silva, and John C Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *science* 290, 5500 (2000), 2319–2323.
[23] Athanasios Theocharidis, Stjin Van Dongen, Anton J Enright, and Tom C Freeman. 2009. Network visualization and analysis of gene expression data using BioLayout Express3D. *Nature protocols* 4, 10 (2009), 1535–1550.
[24] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1225–1234.
[25] Pinar Yanardag and S.V.N. Vishwanathan. 2015. Deep Graph Kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, New York, NY, USA, 1365–1374. https://doi.org/10.1145/2783258.2783417
[26] Ömer Nebil Yaveroğlu, Noël Malod-Dognin, Darren Davis, Zoran Levnajic, Vuk Janjic, Rasa Karapandza, Aleksandar Stojmirovic, and Nataša Pržulj. 2014. Revealing the hidden language of complex networks. *Scientific reports* 4 (2014), 4547.
[27] R. Zafarani and H. Liu. 2009. Social Computing Data Repository at ASU. (2009). http://socialcomputing.asu.edu
[28] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2016. Homophily, Structure, and Content Augmented Network Representation Learning. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 609–618.

RIGHTSLINK