Deep Recursive Network Embedding with Regular Equivalence

Ke Tu[*]
Tsinghua University
tuk15@mails.tsinghua.edu.cn

Peng Cui
Tsinghua University
cuip@tsinghua.edu.cn

Xiao Wang
Tsinghua University
wangxiao007@mail.tsinghua.edu.cn

Philip S. Yu
University of Illinois at Chicago &
Institute for Data Science,
Tsinghua University
psyu@uic.edu

Wenwu Zhu
Tsinghua University
wwzhu@tsinghua.edu.cn

ABSTRACT

Network embedding aims to preserve vertex similarity in an embedding space. Existing approaches usually define the similarity by direct links or common neighborhoods between nodes, i.e. structural equivalence. However, vertexes which reside in different parts of the network may have similar roles or positions, i.e. regular equivalence, which is largely ignored by the literature of network embedding. Regular equivalence is defined in a recursive way that two regularly equivalent vertexes have network neighbors which are also regularly equivalent. Accordingly, we propose a new approach named *Deep Recursive Network Embedding* (DRNE) to learn network embeddings with regular equivalence. More specifically, we propose a layer normalized LSTM to represent each node by aggregating the representations of their neighborhoods in a recursive way. We theoretically prove that some popular and typical centrality measures which are consistent with regular equivalence are optimal solutions of our model. This is also demonstrated by empirical results that the learned node representations can well predict the indexes of regular equivalence and related centrality scores. Furthermore, the learned node representations can be directly used for end applications like structural role classification in networks, and the experimental results show that our method can consistently outperform centrality-based methods and other state-of-the-art network embedding methods.

KEYWORDS

network embedding, regular equivalence, recurrent neural network

ACM Reference Format:
Ke Tu, Peng Cui, Xiao Wang, Philip S. Yu, and Wenwu Zhu. 2018. Deep Recursive Network Embedding with Regular Equivalence. In *KDD 2018: 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3219819.3220068

[*]Tsinghua National Laboratory for Information Science and Technology(TNList)

1 INTRODUCTION

Network embedding [12, 33–35] has aroused considerable research interests in recent years. The fundamental problem is how to preserve the vertex similarity in an embedding space, i.e. if two vertexes are structurally similar in the original network, they should have the similar embedding vectors. There are multiple ways to quantify the similarity of vertexes in a network. The most common one is *structural equivalence* [18]. Two vertexes are structurally equivalent if they share many of the same network neighbors. Most of previous works on network embedding aim to preserve structural equivalence through high-order proximities [33, 35], where network neighbors are extended into high-order neighbors, e.g. direct neighbors, neighbors-of-neighbors, etc.

There are, however, many cases in which vertexes have similar roles or occupy similar positions without any common neighbor. For example, two mothers have the same pattern of connections with a husband and several children. Although the two mothers are not structurally equivalent if they do not have the same relatives, they do share similar roles or positions. These cases lead us to an extended definition of vertex similarity known as *regular equivalence*. Two vertexes are defined to be regularly equivalent if they have network neighbors which are themselves similar (i.e. regularly equivalent) [30]. It is apparent that regular equivalence is a relaxation of structural equivalence. Structural equivalence promises regular equivalence, but the reverse direction does not hold. Comparatively, regular equivalence is more flexible and capable of covering a broad range of network applications related to structural roles or node importance, but is largely ignored by the literature of network embedding.

In order to preserve regular equivalence in network embedding, i.e. two regularly equivalent nodes should have similar embeddings, a straightforward method is to explicitly calculate the regular equivalence of all vertex pairs and require the similarities of node embeddings to approximate their corresponding regular equivalence. But this is infeasible for large-scale networks due to the high complexity of calculating regular equivalence. An alternative is to replace regular equivalence into simpler graph theoretic metrics, such as centrality measures. Although many centrality measures have been designed to characterize the role and importance of a vertex, one centrality can only capture a specific aspect of network role, making it difficult to learn general and task-independent node embeddings. Not to mention that some centrality measures, like betweeness centrality, also bear high computational complexity.

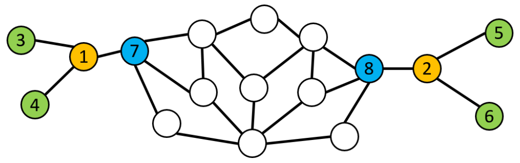


Figure 1: A simple graph to illustrate the rationality of why recursive embedding can preserve regular equivalence. The nodes with the same color are regularly equivalent.

How to effectively and efficiently preserve regular equivalence in network embedding is still an open problem.

As mentioned, the definition of regular equivalence is recursive. This enlightens us to learn network embedding in a recursive way, i.e. the embedding of one node is aggregated by its neighbors' embeddings. In one recursive step (as shown in Figure 1), if nodes 3 and 5, 4 and 6, 7 and 8 are regularly equivalent and thus have similar embeddings already, then nodes 1 and 2 would have similar embeddings, leading to their regular equivalence as true. It is based upon this idea that we propose a novel *Deep Recursive Network Embedding* (DRNE) method. More specifically, we transform the neighbors of a node into an ordered sequence, and propose a layer normalized LSTM (Long Short Term Memory networks) [14] to aggregate the embeddings of neighbors into the embedding of a targeting node in a non-linear way. We theoretically prove that some popular and typical centrality measures are optimal solutions of our model. This is also demonstrated by empirical results that the learned node representations can well preserve pair-wise regular equivalence and predict the values of multiple centrality measures for each node. The learned node representations can be directly used for end applications like structural role classification in networks, and the experimental results show that our method can consistently outperform each single centrality measure, combination of multiple centrality measures and other node representation learning methods.

It is worthwhile to highlight the following contributions of this paper:

- We investigate a novel problem of learning node representations with regular equivalence, which is critical for network analysis and largely ignored in the literature of network representation learning.
- We find an effective and efficient way to incorporate global regular equivalence related information into node representations, and propose a novel deep model DRNE to learn node representations by aggregating neighbors' representations recursively in a non-linear way.
- We theoretically prove that the learned node representations can well preserve pair-wise regular equivalence and reflect several popular and typical node centralities. The empirical results also show the significant advantages of our method over centrality measures as well as other network embedding methods in structural role classification.

The rest of the paper is structured as follows. In the next section, we briefly survey recent related work. Section 3 presents the proposed model in details. Experimental results are presented in section 4. Finally, Section 5 concludes the paper with a brief discussion.

2 RELATED WORK

Recently, network representation learning [12, 23, 33, 39, 40] arouses considerable research interests and an elaborate survey can be found in [8]. Most of the existing network embedding methods are developed along the line of preserving observed pair-wise similarity and structural equivalence. For example, DeepWalk [28] uses random walk to generate sequences of nodes from a network and exploits a language model to learn node representations by treating the sequences as sentences. node2vec [12] extends this idea and propose a biased second order random walk model. LINE [33] optimizes an objective function which aims to preserve both the pairwise similarity and structural equivalence of nodes. More macroscopic structure, the community structure, is incorporated by M-NMF [36] into embedding methods. Furthermore, [35] claims that the underlying structure of the network is highly non-linear and propose a deep auto-encoder model to preserve the first-order and second-order proximities of network structure. Besides, some recent works, such as [15] and [19], affiliate node attribute into the networks and smoothly embed both attribute information and topology structure into a low-dimensional representation. Recently there is a paucity of works related to our targeting problem. For example, RolX [13] enumerates various hand-crafted structural features for nodes and finds the more suited basis vector for this joint feature space. Similarly, struc2vec [29] measures structural similarity by defining a certain form of centrality heuristically. The explicit calculation of pair-wise centrality similarities makes it unscalable. None of these methods preserve regular equivalence while learning representations.

Regular equivalence, as a relaxed notion of structural equivalence, can better capture the structural information. REGE [7] and CATREGE [7] are proposed by searching for an optimal matching between the neighbors of the two vertices iteratively. VertexSim [18] uses the recursive method of linear algebra to construct the measures of similarity. But this is infeasible for large-scale networks due to the high complexity of calculating regular equivalence. Besides, centrality measures are another way to measure the structure information of nodes in networks. A set of centralities [3, 20, 21] have been proposed to study how to capture the structural information better. Since each of them only captures one aspect of structural information, a certain centrality cannot well support different networks and applications. In addition, the hand-crafted manner of designing centrality measures makes them less comprehensive to incorporate regular equivalence related information.

In summary, there is still no sound solution to learning node representations with regular equivalence.

3 DEEP RECURSIVE NETWORK EMBEDDING

In this section, we introduce the proposed method Deep Recursive Network Embedding (DRNE). The framework is shown in Figure 2.

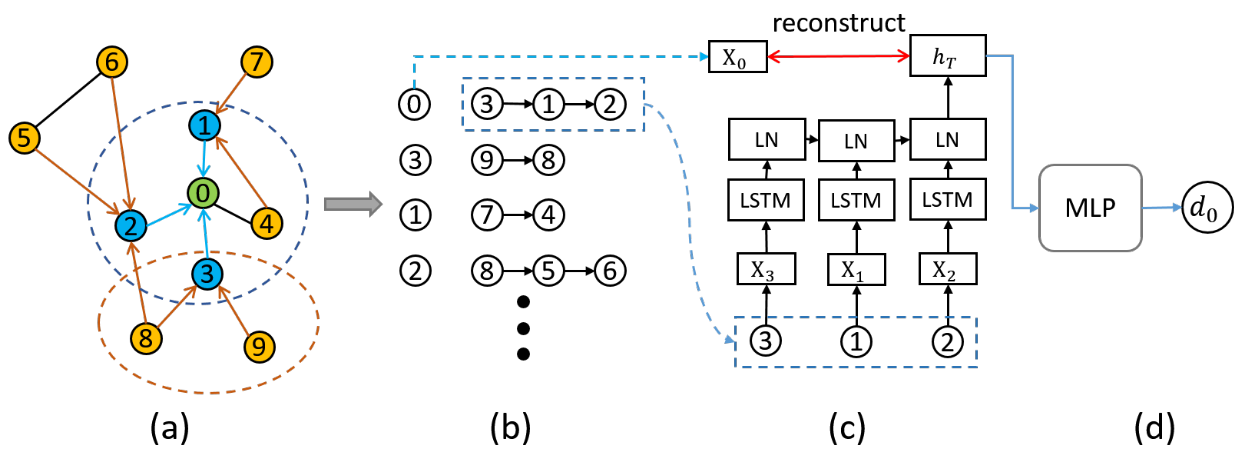


Figure 2: Framework of Deep Recursive Network Embedding (DRNE). (a): Sampling neighborhoods. (b): Sorting neighborhoods by their degrees. (c): Layer-normalized LSTM to aggregate embeddings of neighboring nodes into the embedding of the target node. X_i is the embedding of node i and LN means layer normalization. (d): A Weakly guided regularizer.

3.1 Notations and Definitions

Given a network $G = (V, E)$, where V is the set of nodes and $E \in V \times V$ is the set of edges. For a node $v \in V$, $\mathcal{N}(v) = \{u|(v,u) \in E\}$ is the set of its neighborhoods. The learned embeddings are defined as $\mathbf{X} \in \mathbb{R}^{|V| \times k}$ where k is the dimension and $\mathbf{X}_v \in \mathbb{R}^k$ represents the embedding of node v. We define the degree of node v as $d_v = |\mathcal{N}(v)|$ and function $I(x) = 1$ if $x \geq 0$ otherwise 0. We also give the strict mathematical definition of structural equivalence and regular equivalence.

Definition 3.1 (Structural Equivalence). We denote $s(u) = s(v)$ if nodes u and v are structurally equivalent. Then $s(u) = s(v)$ if and only if $\mathcal{N}(u) = \mathcal{N}(v)$.

Definition 3.2 (Regular Equivalence). We denote $r(u) = r(v)$ if nodes u and v are regularly equivalent. Then $r(u) = r(v)$ if and only if $\{r(i)|i \in \mathcal{N}(u)\} = \{r(j)|j \in \mathcal{N}(u)\}$.

3.2 Recursive Embedding

According to Definition 3.2, we learn node embeddings in a recursive way that the embedding of a target node can be approximated by the aggregation of its neighbors' embeddings. Based on this notion, we design the following loss function:

$$\mathcal{L}_1 = \sum_{v \in V} ||\mathbf{X}_v - Agg(\{\mathbf{X}_u | u \in \mathcal{N}(v)\})||_F^2, \tag{1}$$

where *Agg* is the aggregating function. In one recursive step, the learned embedding of a node can preserve the local structure of its neighbors. By updating the learned representations iteratively, the learned node embeddings can incorporate the their structural information in a global sense, which is consistent with the definition of regular equivalence.

As the underlying structures of real networks are often highly nonlinear [22], we design a deep model, the layer normalized Long Short-Term Memory (ln-LSTM) [2] as the aggregating function. LSTM is known to be effective for modeling sequences. However, the neighbors of a node have no natural ordering in networks. Here we use the degree of nodes as the criterion to sort neighbors into an ordered sequence, mainly because degree is the most efficient measure for neighbor ordering and degree often plays an important role in many graph-theoretic measures, especially those related with structural roles such as PageRank [27] and Katz [25].

Suppose the embeddings of the ordered neighbors are $\{X_1, X_2, ..., X_t, ..., X_T\}$. At each time step t, the hidden state h_t is a function of input embedding X_t at time t and its previous hidden state h_{t-1}, i.e. $h_t = LSTMCell(h_{t-1}, X_t)$. When the embedding sequence is processed by the LSTM Cell recursively from 1 to T, the information of hidden representation h_t will be more and more abundant. h_T can be regarded as the aggregating representation of the neighbors. To learn long-distance correlations in long sequence, The LSTM utilizes gating mechanisms. The forget gate decides what information we are going to throw away from the memory, the input gate along with old memory decides what new information we are going to store in the memory and output gate decides what we are going to output based on the memory. Specifically, The LSTM transition equation *LSTMCell* is the following:

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \cdot [\mathbf{h}_{t-1}, \mathbf{X}_t] + \mathbf{b}_f), \tag{2}$$
$$\mathbf{i}_t = \sigma(\mathbf{W}_i \cdot [\mathbf{h}_{t-1}, \mathbf{X}_t] + \mathbf{b}_i), \tag{3}$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_o \cdot [\mathbf{h}_{t-1}, \mathbf{X}_t] + \mathbf{b}_o), \tag{4}$$
$$\tilde{\mathbf{C}_t} = tanh(\mathbf{W}_C \cdot [\mathbf{h}_{t-1}, \mathbf{X}_t] + \mathbf{b}_C), \tag{5}$$
$$\mathbf{C}_t = \mathbf{f}_t * \mathbf{C}_{t-1} + \mathbf{i}_t * \tilde{\mathbf{C}}_t, \tag{6}$$
$$\mathbf{h}_t = \mathbf{o}_t * tanh(\mathbf{C}_t), \tag{7}$$

where σ is the sigmoid function, $\cdot$ and $*$ represent matrix product and element-wise product respectively, $\mathbf{f}_t$, $\mathbf{i}_t$ and $\mathbf{o}_t$ are forget gate, input gate and output gate respectively and C_t is the cell state. $\mathbf{W}_*$ and $\mathbf{b}_*$ are learned parameters.

Besides, in order to avoid the problems of exploding or vanishing gradients [14] with the long sequences as inputs, we also introduced

Layer Normalization [2]. The layer normalized LSTM makes it invariant to re-scaling all of the summed inputs. It results in much more stable dynamics. Particularly, it recenters and re-scales the cell state C_t after Equation 6 using the extra normalization like follows:

$$C'_t = \frac{g}{\Sigma_t} * (C_t - \mu_t), \tag{8}$$

where $\mu_t = 1/k \sum_{i=1}^{k} C_{ti}$ and $\Sigma_t = \sqrt{1/k \sum_{i=1}^{k} (C_{ti} - \mu_t)^2}$ are the mean and variance of C_t , and g is gain parameter scaling the normalized activation.

3.3 Regularization

$\mathcal{L}_1$ defined in Equation 1 expresses the recursive embedding process according to Definition 3.2 without any other constraints. It has such a strong expressive power that multiple solutions can be derived as long as they satisfy the given recursive process. It is risky for this model to degenerate to the trivial solution with all the embeddings being 0. To avoid the trivial solution, we use node degree as the weakly guided information and impose a constraint that the learned embedding of a node should be able to approximate the degree of the node. Accordingly, we design the following regularizer:

$$\mathcal{L}_{reg} = \sum_{v \in V} \|\log(d_v + 1) - MLP(Agg(\{\mathbf{X}_u | u \in \mathcal{N}(v)\}))\|_F^2, \tag{9}$$

where d_v is the degree of node v, MLP is single-layer multilayer perceptron with rectified linear unit (ReLU) [11] activation which is defined as $ReLU(x) = max(0, x)$. In total, we minimize the objective function by combining reconstruction loss of Equation 1 and the regularizer of Equation 9:

$$\mathcal{L} = \mathcal{L}_1 + \lambda \mathcal{L}_{reg}, \tag{10}$$

where the λ is the weight of regularizer. Note that here the degree information is not used as supervise information of network embedding. Instead, it eventually plays an auxiliary role to induce the solution to be far from the trivial solution. Thus, the λ here is quite a small value.

Neighborhood Sampling. In real networks, the degree of nodes often follow heavy-tailed distribution [9], i.e. a small number of nodes have very high degree while the majority of nodes have very small degree. In order to improve the efficiency, we downsample the neighbors of nodes with large degree before inputing them into the ln-LSTM. Specifically, we set an upper bound of the number of the neighbors S. If the number of neighbors exceeds the upper bound S, we downsample them into S different nodes. Figure 2 (a) and (b) shows an example on how to sample from neighborhoods. In power-law networks, the nodes with large degree carry more unique structural information than the common nodes with small degree. Thus we design a biased sampling strategy to keep the nodes with large degree by setting the sampling probability $P(v)$ of node v being proportional to its degree d_v, $P(v) \propto d_v$.

3.4 Optimization

To optimize the aforementioned model, the goal is to minimize the overall loss $\mathcal{L}$ as a function of the neural network parameters set θ and the embeddings X. Adam [16] is used to optimize these parameters. The derivatives are estimated using the BackPropagation Through Time (BPTT) algorithm [37]. The learning rate α for Adam is initially set to 0.0025 at the beginning of the training.

3.5 Theoretical Analysis

Our method is designed according to the recursive definition of regular equivalence. It is intuitive that the learned embedding should be able to preserve regular equivalence and the empirical results in section 4.3 also demonstrate it. As an additional evidence, here we further theoretically prove that the resulted embeddings of our method can well reflect several typical and common centrality measures which are closely related with regular equivalence. Without loss of generality, we ignore the regularizer term in Equation 9 which is used for avoiding trivial solution.

Theorem 3.3. *Degree centrality, eigenvector centrality [5], PageRank centrality [27] are three optimal solutions of our model respectively.*

To prove Theorem 3.3, we first prove following lemmas:

Lemma 3.4. *For any computable function, there exists a finite recurrent neural network (RNN) [24] that can compute it.*

Proof. This is a direct consequence of Theorem 1 in [32]. □

Theorem 3.5. *If the centrality $C(v)$ of node v satisfies that $C(v) = \sum_{u \in \mathcal{N}(v)} F(u)C(u)$ and $F(v) = f(\{F(u), u \in \mathcal{N}(v)\})$ where f is any computable function, then $C(v)$ is one of the optimal solutions of our model.*

Proof. For simplicity, we suppose that all the activation function of LSTM are linear activation. We prove this lemma by proving that there exists a parameter setting $\{\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_o, \mathbf{W}_C, \mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_o, \mathbf{b}_C\}$ in Equation 2-5 such that the node embedding $\mathbf{X}_u = [F(u), C(u)]$ is a fixed point. We directly construct this parameter settings. Suppose $\mathbf{W}_{a,i}$ donates the i-th row of $\mathbf{W}_a$. With the input of sequence $\{[F(u), C(u)], u \in \mathcal{N}(v)\}$, set $\mathbf{W}_{f,2}$ and $\mathbf{W}_{o,2}$ as $[0, 0]$, $\mathbf{W}_{i,2}$ as $[1, 0]$, $\mathbf{W}_{C,2}$ as $[0, 1]$, $\mathbf{b}_{f,2}$ and $\mathbf{b}_{o,2}$ as 1, $\mathbf{b}_{i,2}$ and $\mathbf{b}_{C,2}$as 0, then we can easily obtain $\mathbf{h}_{t,2} = \mathbf{o}_{f,2} * \mathbf{C}_{t,2} = \mathbf{C}_{t,2} = \mathbf{f}_{t,2} * \mathbf{C}_{t-1,2} + \mathbf{i}_{t,2} * \tilde{\mathbf{C}}_{t,2} = \mathbf{C}_{t-1,2} + F(t) * C(t)$. Thus $\mathbf{h}_{T,2} = \sum_{u \in \mathcal{N}(v)} F(u)C(u) = C(v)$ where T is the length of the input sequence. Besides, by Lemma 3.4, there exists a parameter setting $\{\mathbf{W}'_f, \mathbf{W}'_i, \mathbf{W}'_o, \mathbf{W}'_C, \mathbf{b}'_f, \mathbf{b}'_i, \mathbf{b}'_o, \mathbf{b}'_C\}$ to approximate f. By set $\mathbf{W}_{f,1}$ as $[\mathbf{W}'_f, 0]$, $\mathbf{W}_{o,1}$ as $[\mathbf{W}'_o, 0]$ and so on, we can obtain that $\mathbf{h}_{T,1} = f(\{F(u), u \in \mathcal{N}(v)\}) = F(v)$. Thus $\mathbf{h}_T = [F(v), C(v)]$ and the node embedding $\mathbf{X}_v = [F(v), C(v)]$ is a fixed point. This completes the proof. □

By the definitions of centralities in Table 1 with $(F(v), f(\{x_i\}))$, we can easily obtain that degree centrality, eigenvector centrality, PageRank centrality satisfy the condition of Theorem 3.5, which completes the proof of Theorem 3.3.

According to Theorem 3.3, for any graph, there exists such a parameter setting of our proposed method that the learned embeddings can be one of the three centralities. This demonstrates the expressive power of our method in capturing different aspects of network structural information that are related with regular equivalence.

Table 1: Definition of centralities.

Centrality	Definition $C(v)$	$F(v)$	$f(\{x_i\})$
Degree	$d_v = \sum_{u \in \mathcal{N}(v)} I(d_u)$	$1/d_v$	$1/(\sum I(x_i))$
Eigenvector	$1/\lambda * \sum_{u \in \mathcal{N}(v)} C(u)$	$1/\lambda$	mean
PageRank	$\sum_{u \in \mathcal{N}(v)} 1/d_u * C(u)$	$1/d_v$	$1/(\sum I(x_i))$

3.6 Analysis and Discussions

In this section, we present the out-of-sample extension and the complexity analysis.

3.6.1 Out-of-sample Extension. For a newly arrived node v, if its connections to the existing nodes are known, we can directly feed the embeddings of its neighbors into the aggregating function and obtain the aggregated representation as the embedding of the node with Equation 1. The complexity for such a procedure is $O(d_v k)$, where k is the length of embeddings and d_v is the degree of node v.

3.6.2 Complexity Analysis. During the training procedure, for a single node v in each iteration, the time complexity of calculating gradients and updating parameters is $O(d_v k^2)$, where d_v is the degree of node v, and k is the length of embeddings. Due to the sampling process, the degree d_v will not exceed the limited number S. Thus the overall training complexity is $O(|V|Sk^2 I)$ where I is the number of iterations. The length of embeddings k is usually set as a small number such as 32, 64, 128. The limited number S is set as 300 in this work. And the number of iterations I is normally a small number but independent with the number of nodes $|V|$. Therefore, the complexity of training procedure is linear to the number of nodes $|V|$.

4 EXPERIMENTS

In this section, We evaluate our method on different benchmarks to prove its efficacy.

4.1 Baselines and Parameter Settings

- DeepWalk [28]: This algorithm learns node representations by modeling a stream of short random walks. We set window size as 10, walk length as 40 and walks per node as 40.
- LINE [33]: This algorithm learns feature representations in two separate phases which preserve the first-order and second-order proximities separately. The number of negative samples K = 5 and the mini-batch size of the stochastic gradient descent is set to 1.
- node2vec [12]: Node2vec extends DeepWalk by proposing a biased second order random walk model, making it more flexible when generating the context of a node. The basic parameter settings are the same as DeepWalk. We set p as 1 and q as 2 in node2vec to discover more neighborhood information.
- struc2vec [29]: Struc2vec learns latent representations for the structural identity of nodes. Due to its high computational complexity, we use the combination of all optimizations proposed in the paper for large networks.
- Centralities: Four popular centralities, i.e. Closeness centrality [26], Betweenness centrality [4], Eigenvector centrality [6] and K-core [17], are used to measure node importance. Besides, We concatenate all of the four previous centralities into one vector as another baseline (*Combined*).

For our model, we generally set the length of embeddings k as 16, the weight of the regularizer λ is 1, and the limited neighborhood number S is 300. We will show that our model is not very sensitive to these parameters in Section 4.5.

4.2 Network Visualization

In this section, we visualize the learned embeddings to intuitively show that our model preserves regular equivalence.

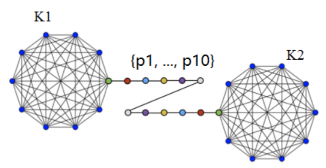
Visualization on Barbell Network. The graph shown in Figure 3(a) consists of two complete graphs K_1 and K_2 connected by a path graph P of length 10. Each complete graph has 10 nodes. $\{p_1, p_2,...,p_{10}\}$ denotes the nodes of path graph P and p_i connects p_{i+1} for $i = 1, 2, ..., 9$. p_1 connects node b_1 from K_1 and p_2 connects b_2 from K_2. As a result of the symmetry of barbell graph, there are many node pairs with identical structural roles. All nodes from $\{K_1/\{b_1\}\} \cup \{K_2/\{b_2\}\}$ should be regularly equivalent. Besides, all node pairs $(p_i, p_{11-i}), i = 1, ...10$ and (b_1, b_2) should also be regularly equivalent. The barbell graph is illustrated in Figure 3(a) and the nodes are regularly equivalent.

Figure 3 shows the learned latent representations by different baselines. From the results, we have following observations:

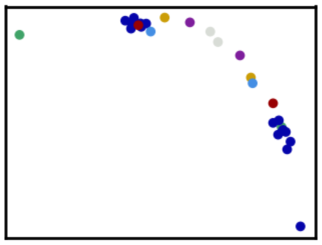
- We can see that the nodes from the two complete graph K_1 and K_2 have a large margin in the embedding spaces generated by DeepWalk, LINE and node2vec. DeepWalk and LINE fail to preserve the structural importance, which is natural since they only consider the pair-wise relationships. Although node2vec could incorporate some local neighborhood information, the algorithm still tend to focus on pairwise relationships.
- struc2vec and our proposed model DRNE achieve similar results in this task. Both of them place nodes with similar structural roles close in latent space. However, struc2vec only preserves the similarities of local K-neighborhoods which cannot reflect global structural information. Since the local and global structural informations are fuzzy in such a small symmetric graph, the performance of DRNE and struc2vec cannot be differentiated. We will prove it in the next section.

Visualization on Karate Network. Karate network network shown in Figure 4(a) is a well-known social network of a university karate club. It captures 34 members and 78 pair-wise links between members who interact outside the club. The learned latent representations are shown in Figure 4. The color of nodes represents the value of k-core, which measures the spreading efficiency of a node and shows the global position of a node in the networks. The colors red, light blue, light yellow, draw blue represent the k-core value from 1 to 4 respectively.

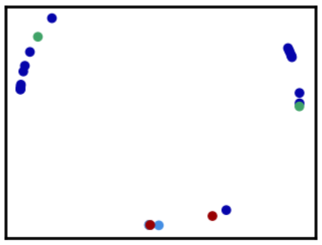
From Figure 4, we can see that the results of DeepWalk, LINE and node2vec are similar to those on the barbell graph. They cannot distinguish the difference of global structures and mix them together. The struc2vec confuses nodes with different k-core values in the right part of Figure 4(e). It is reasonable since struc2vec



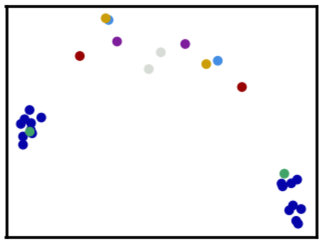
(a) barbell graph



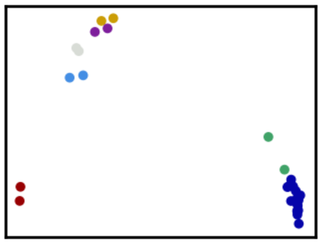
(b) DeepWalk



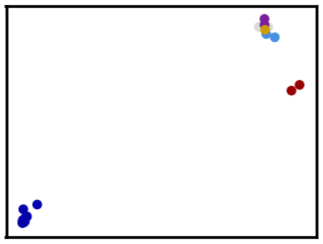
(c) LINE



(d) node2vec

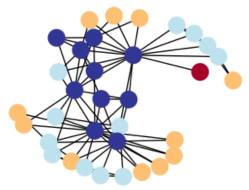


(e) struc2vec

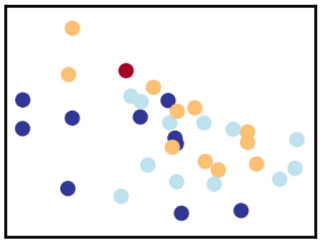


(f) DRNE

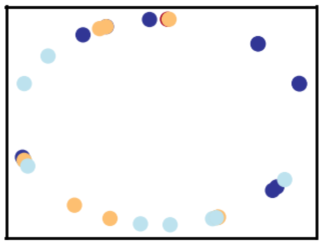
Figure 3: Visualization on Barbell graph. (a) Barbell graph. Latent representations in $\mathbb{R}^2$ learned by (b) DeepWalk, (c) LINE, (d)node2vec, (e) struc2vec and (f) our proposed method DRNE.



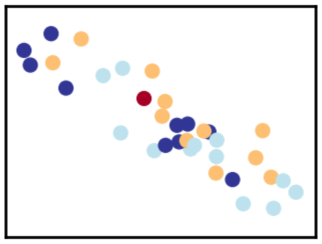
(a) karate graph



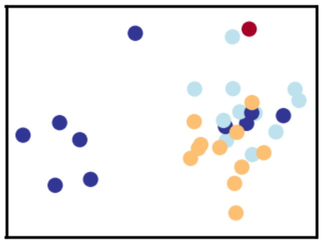
(b) DeepWalk



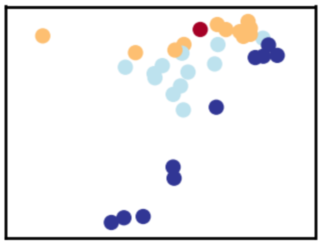
(c) LINE



(d) node2vec



(e) struc2vec



(f) DRNE

Figure 4: Visualization on karate graph. (a) Karate graph. Latent representations in $\mathbb{R}^2$ learned by (b) DeepWalk, (c) LINE, (d)node2vec, (e) struc2vec and (f) DRNE.

preserves only the similarities of local neighborhoods and cannot capture the global structural roles. Our proposed model separates the nodes with equivalent k-core values approximatively. From left-top to right-bottom in Figure 4(f), the k-core value increases gradually. This result demonstrates that our model can preserve both the local neighborhood information and the global structural roles information of the target networks.

Table 2: The MSE value of predicting centralities on Jazz dataset ($*10^{-2}$)

centrality	closeness	betweenness	eignvector	k-core
DeepWalk	0.6016	3.7188	2.1543	13.2755
LINE	0.5153	4.3919	1.5072	15.8179
node2vec	1.0489	3.4065	3.9436	39.2156
struc2vec	0.2365	0.25371	1.0544	9.0858
DRNE	**0.1909**	**0.1261**	**0.5267**	**5.5683**

Table 3: The MSE value of predicting centralities on BlogCatalog dataset ($*10^{-2}$)

centrality	closeness	betweenness	eignvector	k-core
DeepWalk	0.2982	1.7836	1.1194	19.7016
LINE	0.3979	1.8425	1.5167	34.9079
node2vec	0.3573	1.6958	1.1432	24.1704
struc2vec	0.2947	1.6018	1.0445	25.3047
DRNE	**0.1101**	**0.6676**	**0.3108**	**7.7210**

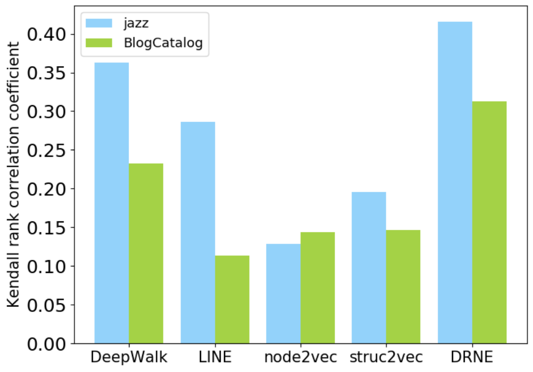


Figure 5: Kendall rank correlation coefficient by fitting regular equivalence on Jazz and BlogCatalog dataset.

4.3 Regular Equivalence Prediction

In this section, we evaluate whether the learned embeddings preserve regular equivalence on two real world datasets, i.e. Jazz [10] and BlogCatalog [38]:

- Jazz [10]: This dataset is the collaboration network between Jazz musicians in 2003. Each edge denotes that two musicians have played together in a band. It contains 198 nodes and 2742 edges.
- BlogCatalog [38]: This is the data set crawled from BlogCatalog. BlogCatalog is the social blog directory which manages the bloggers and their blogs. The edges denote two bloggers are friends in BlogCatalog. It contains 10312 nodes and 333983 edges.

Here we set up two tasks, including regular equivalence prediction and centrality score prediction.

For the first task, we use a commonly used regular equivalence-based similarity measure method Vertex Similarity in Network (VS) [18] as ground truth. The pairwise similarity of two nodes is measured by the inner product of their embeddings. We rank node pairs by their similarities and compare the result rank with the rank by VS. Kendall rank correlation coefficient [1] is used to measure the correlation of these two ranks. It is defined as follows:

$$\tau(x, y) = \frac{|\{(i,j)|(x_i - x_j)(y_i - y_j) > 0\}| - |\{(i,j)|(x_i - x_j)(y_i - y_j) < 0\}|}{n(n-1)/2}. \tag{11}$$

The results are shown in Figure 5. We can see that our method outperforms all the baselines. It demonstrates our method can preserve regular equivalence while learning representations.

For the second task, we calculate the four mentioned centralities in Section 4.1 as ground truth. Then we randomly hide 20 percentage of nodes and use the remaining nodes to train a linear regression model to predict the centrality scores based on node embeddings. After training, we use the regression model to predict the centralities of the held-out nodes. Mean Square Error (MSE) is used to evaluate the performance. The MSE is defined as follows:

$$MSE(Y, \hat{Y}) = \frac{1}{n}\sum_{i=1}^{n}(Y - \hat{Y})^2, \tag{12}$$

where Y is the observed value and $\hat{Y}$ is the predicted value. Since the scales of different centralities are significantly different, we rescale the MSE value by dividing it by the corresponding averaged centrality values of all nodes.

The results are shown in Table 2 and Table 3. The observations are illustrated as follows:

- Our method achieves significant improvements over the baselines in predicting all the four centralities. It demonstrates that our method has powerful representation ability to preserve structural role related information.
- Compared with other baselines, struc2vec achieves higher improvements on small and dense Jazz dataset than on large and sparse BlogCatalog dataset. Because struc2vec can only capture the structural information of local neighbors, which is not sufficient to describe the regular equivalence of two nodes in a large and sparse network. Comparatively, our DRNE is able to capture global structural information due to the recursive embedding procedure. The significant difference of DRNE and struc2vec in predicting k-core can fully demonstrate this point, considering that k-core is designed to reflect the global position of a node in networks.

4.4 Structural Role Classification

Classification [31] is a common and important network application. In this structural role based classification task, the labels for nodes are more related to their structural information than to the labels of their adjacent nodes. In this section, we evaluate the ability of learned embeddings in predicting the structural roles of nodes in a given network. We compare our model with not only the state-of-the-art network embedding methods but also four popular centrality

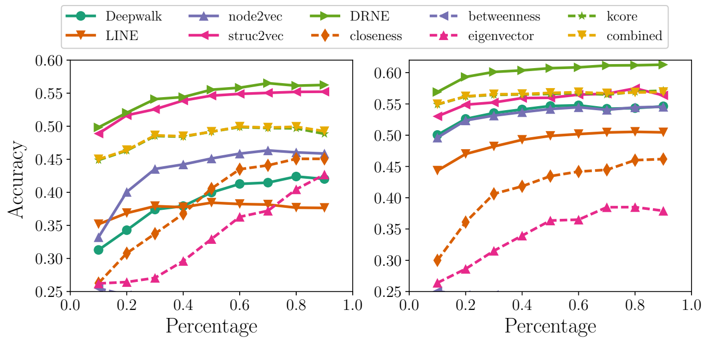


Figure 6: Average accuracy for multi-class node classification in air-traffic network. Left: Europe air-traffic. Right: American air-traffic

measures, including closeness centrality, betweenness centrality, eigenvector centrality and k-core.

We evaluate these methods in two air-traffic networks, i.e. European air-traffic network and American air-traffic network [29]. The nodes represent airports and the edges indicate the existence of direct flights. The total number of landings plus takeoffs or the total number of people that pass the airports can be used to measure their activities that reflects their structural roles in the flight networks. One of four possible labels is assigned for each airport by splitting their activity distribution uniformly.

After obtaining the embeddings of the airports, we train a logistic regression to predict the labels based on the embeddings. We randomly sample 10% to 90% of the nodes as the training samples and use the left nodes to test the performance. Averaged accuracy is used to evaluate the performance. The results are shown in Figure 6. The curves with solid line indicate network embedding methods and the other curves with dashed line indicate centrality methods.

From the results, we have following observations:

- Our model DRNE outperforms all the other baselines on these two datasets. It demonstrates the effectiveness of our proposed method. Moreover, Our model outperforms the combined centralities, it indicates that DRNE can preserve more comprehensive regular equivalence related information than manually defined centrality measures.
- The relative improvement of our method over the best baseline struc2vec in American air-traffic network is more obvious than that in European air-traffic network. Note that struc2vec can only preserve the similarity of local K-neighbors, and cannot capture the global structural information in the larger American air-traffic network. This demonstrates the importance of preserving global structural information as in DRNE.

4.5 Parameter Sensitivity

In this section, we evaluate the scalability and how parameters influence the performance. Especially, we evaluate the effect of the embedding dimension k, the weight of regularizer λ and the upper bound of neighborhood size S. For brevity, we report the results in classification task with flight datasets.

4.5.1 Effect of parameter k. We show how the dimension of embedding space k affects the performance in Figure 7(a). Since higher embedding dimensions can embody more information, the performance raises firstly when the number of embedding dimension increases. Then after the dimension k exceeds 16, the performance becomes stable. This demonstrates that our algorithm is insensitive to embedding dimensions.

4.5.2 Effect of parameter λ. Figure 7(b) shows how the weight of regularizer λ affects the performance. We select the λ from $\{0.1, 0.5, 1.0, 1.5, 2.0\}$. The performance curve is relatively stable, demonstrating that our model is not sensitive to this parameter. This also support our notion that the degree information is used to avoid trivial solution, rather than supervise information for learning embeddings.

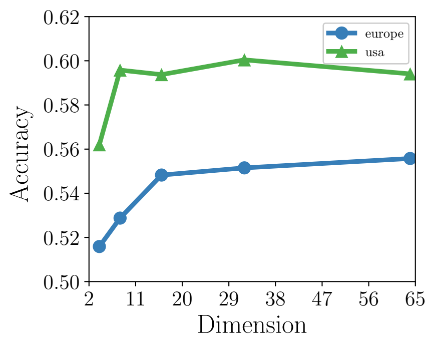
4.5.3 Effect of parameter S. The parameter S controls the maximum number of neighbors when sampling. As shown in Figure 7(c), when this parameter increases in early range, the performance becomes better since more neighborhoods contains more structural informations. However, the performance drops when the parameter S is too large. The main reason is that the gradients explode or vanish when the sequences are too long in an LSTM model. The setting of parameter S needs to trade-off between the obtained information and the capacity of optimizer. In practice, we choose the parameter as 300 so that we can obtain enough information within the capacity of the optimizer.

4.6 Scalability

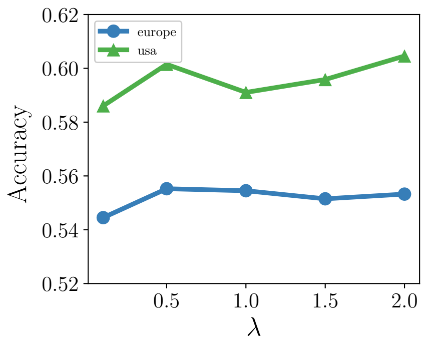
To testify the scalability, we test training time per epoch. The result is shown in Figure 8. The parameters of struc2vec are set as the default value with all proposed optimizations reported in [29]. We can observe that the training time of our model scales linearly with the number of nodes, and the slope of the curve is close to 1 (the bottom dashed line). struc2vec scales super-linearly (closer to $n^{1.5}$), and the slope of the curve is close to 1.5 (the top dashed line). Our model DRNE is much faster than struc2vec by several order of magnitude. This result conforms to the complexity analysis in Section 3.6.2 and proves that our proposed methods is linear to the number of nodes and can scale to large graphs.

5 CONCLUSION

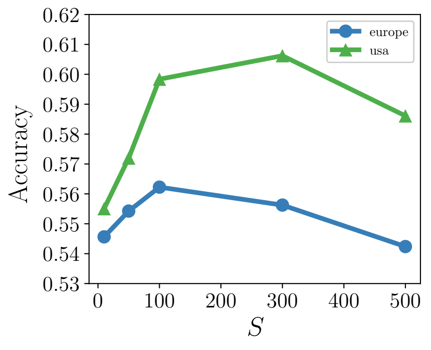
In this paper, we propose a novel deep model to learn node representations with regular equivalence in networks. Assuming that the regular equivalence information of a node has been encoded by the representations of its neighboring nodes, we propose a layer normalized LSTM model to learn node embeddings recursively. For a given node, the structural importance information of far-distance nodes can be recursively propagated to its neighbor nodes and thus can be embedded into its embeddings. Therefore, the learned node embeddings can reflect their structural information in a global sense, which is consistent with regular equivalence and centrality measures. The empirical results demonstrate that our method can significantly and consistently outperform the state-of-the-art algorithms.



(a) k



(b) λ



(c) S

Figure 7: Parameter Sensitivity w.r.t. the dimension of embeddings k, the regularizer weights λ and the upper bound of the number of neighbors S.

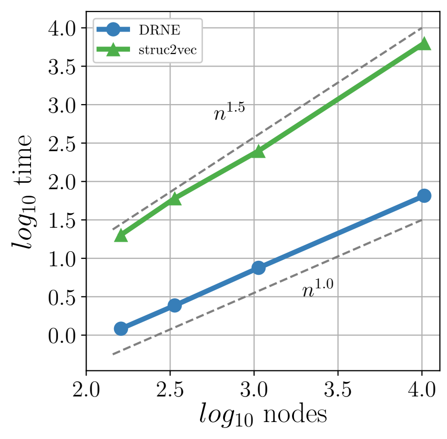


Figure 8: Average training time of struc2vec and DRNE.

6 ACKNOWLEDGMENTS

This work was supported in part by National Program on Key Basic Research Project No. 2015CB352300, National Natural Science Foundation of China Major Project No. U1611461; National Natural Science Foundation of China No. 61772304, 61521002, 61531006, 61702296; NSF IIS-1526499, IIS-1763325, and CNS-1626432, and NSFC 61672313. Thanks for the research fund of Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology, and the Young Elite Scientist Sponsorship Program by CAST. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] Hervé Abdi. 2007. The Kendall rank correlation coefficient. *Encyclopedia of Measurement and Statistics. Sage, Thousand Oaks, CA* (2007), 508–510.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[3] Joonhyun Bae and Sangwook Kim. 2014. Identifying and ranking influential spreaders in complex networks by neighborhood coreness. *Physica A: Statistical Mechanics and its Applications* 395 (2014), 549–559.

[4] Marc Barthelemy. 2004. Betweenness centrality in large complex networks. *The European Physical Journal B-Condensed Matter and Complex Systems* 38, 2 (2004), 163–168.

[5] Phillip Bonacich. 2007. Some unique properties of eigenvector centrality. *Social networks* 29, 4 (2007), 555–564.

[6] Phillip Bonacich and Paulette Lloyd. 2015. Eigenvector centrality and structural zeroes and ones: When is a neighbor not a neighbor? *Social Networks* 43 (2015), 86–90.

[7] Stephen P Borgatti and Martin G Everett. 1993. Two algorithms for computing regular equivalence. *Social networks* 15, 4 (1993), 361–376.

[8] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2017. A Survey on Network Embedding. *arXiv preprint arXiv:1711.08752* (2017).

[9] Young-Ho Eom and Hang-Hyun Jo. 2015. Tail-scope: Using friends to estimate heavy tails of degree distributions in large-scale complex networks. *Scientific reports* 5 (2015).

[10] Pablo M Gleiser and Leon Danon. 2003. Community structure in jazz. *Advances in complex systems* 6, 04 (2003), 565–573.

[11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 315–323.

[12] 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.

[13] Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. 2012. Rolx: structural role extraction & mining in large graphs. In *KDD*. ACM, 1231–1239.

[14] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[15] Xiao Huang, Jundong Li, and Xia Hu. 2017. Label informed attributed network embedding. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 731–739.

[16] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[17] Maksim Kitsak, Lazaros K Gallos, Shlomo Havlin, Fredrik Liljeros, Lev Muchnik, H Eugene Stanley, and Hernán A Makse. 2010. Identification of influential spreaders in complex networks. *arXiv preprint arXiv:1001.5285* (2010).

[18] Elizabeth A Leicht, Petter Holme, and Mark EJ Newman. 2006. Vertex similarity in networks. *Physical Review E* 73, 2 (2006), 026120.

[19] Jundong Li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. 2017. Attributed network embedding for learning in a dynamic environment. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 387–396.

[20] Jian-Hong Lin, Qiang Guo, Wen-Zhao Dong, Li-Ying Tang, and Jian-Guo Liu. 2014. Identifying the node spreading influence with largest k-core values. *Physics Letters A* 378, 45 (2014), 3279–3284.

[21] Linyuan Lü, Tao Zhou, Qian-Ming Zhang, and H Eugene Stanley. 2016. The H-index of a network node and its relation to degree and coreness. *Nature communications* 7 (2016), 10168.

[22] Dijun Luo, Feiping Nie, Heng Huang, and Chris H Ding. 2011. Cauchy graph embedding. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. 553–560.

[23] Jianxin Ma, Peng Cui, and Wenwu Zhu. 2018. DepthLGP: Learning Embeddings of Out-of-Sample Nodes in Dynamic Networks. (2018).

[24] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

[25] Eisha Nathan and David A Bader. 2017. A Dynamic Algorithm for Updating Katz Centrality in Graphs. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*. ACM, 149–154.

[26] Kazuya Okamoto, Wei Chen, and Xiang-Yang Li. 2008. Ranking of closeness centrality for large-scale social networks. *Lecture Notes in Computer Science* 5059 (2008), 186–195.

[27] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.

[28] 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.

[29] 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.

[30] Ryan A Rossi and Nesreen K Ahmed. 2015. Role discovery in networks. *IEEE Transactions on Knowledge and Data Engineering* 27, 4 (2015), 1112–1131.

[31] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93.

[32] Hava T Siegelmann and Eduardo D Sontag. 1995. On the computational power of neural nets. *Journal of computer and system sciences* 50, 1 (1995), 132–150.

[33] 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*. International World Wide Web Conferences Steering Committee, 1067–1077.

[34] Ke Tu, Peng Cui, Xiao Wang, Fei Wang, and Wenwu Zhu. 2017. Structural Deep Embedding for Hyper-Networks. *arXiv preprint arXiv:1711.10146* (2017).

[35] 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.

[36] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community Preserving Network Embedding.. In *AAAI*. 203–209.

[37] Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proc. IEEE* 78, 10 (1990), 1550–1560.

[38] Reza Zafarani and Huan Liu. 2009. Social computing data repository at ASU. (2009).

[39] Ziwei Zhang, Peng Cui, Jian Pei, Xiao Wang, and Wenwu Zhu. 2017. TIMERS: Error-Bounded SVD Restart on Dynamic Networks. *arXiv preprint arXiv:1711.09541* (2017).

[40] Dingyuan Zhu, Peng Cui, Ziwei Zhang, Jian Pei, and Wenwu Zhu. 2018. High-order Proximity Preserved Embedding For Dynamic Networks. *IEEE Transactions on Knowledge and Data Engineering* (2018).