deep neural network for learning graph representations

Conference Paper · February 2016

CITATIONS
241

READS
1,650

1 author:

Shaosheng Cao
Ant Financial
5 PUBLICATIONS **417** CITATIONS

SEE PROFILE

Deep Neural Networks for Learning Graph Representations

Shaosheng Cao
Xidian University
shelsoncao@gmail.com

Wei Lu
Singapore University of Technology and Design
luwei@sutd.edu.sg

Qiongkai Xu
Australian National University
National ICT Australia - CRL
Qiongkai.Xu@nicta.com.au

Abstract

In this paper, we propose a novel model for learning graph representations, which generates a low-dimensional vector representation for each vertex by capturing the graph structural information. Different from other previous research efforts, we adopt a random surfing model to capture graph structural information directly, instead of using the sampling-based method for generating linear sequences proposed by Perozzi et al. (2014). The advantages of our approach will be illustrated from both theorical and empirical perspectives. We also give a new perspective for the matrix factorization method proposed by Levy and Goldberg (2014), in which the pointwise mutual information (PMI) matrix is considered as an analytical solution to the objective function of the skip-gram model with negative sampling proposed by Mikolov et al. (2013). Unlike their approach which involves the use of the SVD for finding the low-dimensitonal projections from the PMI matrix, however, the stacked denoising autoencoder is introduced in our model to extract complex features and model non-linearities. To demonstrate the effectiveness of our model, we conduct experiments on clustering and visualization tasks, employing the learned vertex representations as features. Empirical results on datasets of varying sizes show that our model outperforms other stat-of-the-art models in such tasks.

1 Introduction

Graphs are common representations used for information management in many real-world problems. For example, in protein network research, mining protein complexes from protein-protein interaction networks plays an important role in the description of functional diversity of homologous proteins and serves valuable evolutionary insights, which is essentially a clustering problem. It is therefore crucial to develop automated algorithms to extract useful, deep information from graphs. One of effective means towards organizing such information associated with the potentially large and complex graph is to learn the *graph representations*, which assign to each vertex of the graph a low-dimensional dense vector representation, encoding meaningful information conveyed by the graph.

Recently, there has been significant interest in the work of learning word embeddings (Bullinaria and Levy 2007; Mikolov et al. 2013). Their goal is to learn for each natural language word a low-dimensional vector representation based on their contexts, from a large amount of natural language texts. Such works can be regarded as learning representations for linear sequences, since the corpora consist of linear structures which are natural language word sequences. The resulting compact, low-dimensional vector representations are believed to be able to capture rich semantic information and are proven to be useful for various natural language processing tasks. While efficient and effective methods for learning good representations for linear structures have been identified (Mikolov et al. 2013; Pennington, Socher, and Manning 2014), dealing with general graph structures with rich topologies is more complicated. To this end, one natural approach is to identify effective means to cast the task of learning vertex representations for general graph structures into the task of learning representations from linear structures. DeepWalk proposed by Perozzi et al. (2014) presented an idea to transform *unweighted* graphs into collections of linear sequences by a uniform sampling method known as truncated random walk. In their approach, sampled vertex sequences characterize the connections between vertices in a graph. This step can be understood as a process for converting a general graph structure into a large collection of linear structures. Next, they utilized the skip-gram model proposed by Mikolov et al. (2013) to learn low-dimensional representations for vertices from such linear structures. The learned vertex representations were shown to be effective across a few tasks, outperforming several previous approaches such as spectral clustering and modularity method (Tang and Liu 2009a; 2009b; 2011; Macskassy and Provost 2003).

While such an approach for learning vertex representations of a unweighted graph is effective, two important issues remain to be answered. First, how to accurately and more directly capture the graph structural information for *weighted* graphs? Second, is there a better method of representing vertices of linear structures? To answer the first question, we design a random surfing model suitable for weighted graphs, which can directly yield a probabilistic co-occurrence matrix. Such a matrix is similar to the co-occurrence matrix obtained by sampling linear sequences

from graphs (Bullinaria and Levy 2007), but no sampling process is required in our approach. To answer the second question, we first revisit one popular existing method used for learning vertex representations for linear structures. A recent study by Levy and Goldberg (2014) showed that optimizing the objective function associated with the skip-gram with negative sampling method (Mikolov et al. 2013) has a intrinsic relation with factorzing a shifted *positive pointwise mutual information* (PPMI) matrix (Bullinaria and Levy 2007) of the words and their contexts. Specifically, they showed that it was possible to use the standard *singular value decomposition* (SVD) method to factorize the PPMI matrix to induce the vertex/word representations from the decomposed matrices. Our recent approach called GraRep (Cao, Lu, and Xu 2015) has been shown to achieve good empirical results on the task of learning graph representations. However, the approach employs SVD for performing linear dimension reduction, while better non-linear dimension reduction techniques were not explored.

In this paper, we give a new perspective to the work of Levy and Goldberg (2014). We argue that the original PPMI matrix itself is an explicit representation matrix of the graph, and the SVD step essentially plays its role as a dimension reduction toolbox. While the SVD step for inducing the final word representations was shown effective, Levy et al. (2015) also demonstrated the effectiveness of using the PPMI matrix itself as the word representations. Interestingly, as shown by the authors, the representations learned from the SVD method cannot completely outperform the representations from the PPMI matrix itself (Levy, Goldberg, and Dagan 2015). Since our final goal is to learn good vertex representations that are effective in capturing the graph's information, it is essential to investigate better ways of recovering the vertex representations from the PPMI matrix, where potentially complex, non-linear relations amongst different vertices can be captured.

Deep learning sheds light on the path of modeling non-linear complex phenomena, which has many successful applications in different domains, such as speech recognition (Dahl et al. 2012) and computer vision (Krizhevsky, Sutskever, and Hinton 2012). Deep neural networks (DNN), *e.g.,* the stacked autoencoders, can be regarded as an effective method for learning high level abstractions from low level features. This process essentially performs dimension reduction that maps data from a high dimensional space into a lower dimensional space. Unlike the (truncated) SVD-based dimension reduction method, which maps from the original representation space to a new space with a lower rank through a *linear* projection, deep neural networks such as stacked auto-encoders can learn projections which are highly *non-linear*. In fact, a recent work by Tian et al. (2014) used sparse autoencoders to replace the eigenvalue decomposition step of spectral clustering in the clustering task, and they achieved a significant improvement. Motivated by their work, we also investigate the effectiveness of using alternative deep learning-based approaches for learning low-dimensional representations from the original data representations. Different from their work, we aim at learning vertex representations for general graphs, and do not exclusively focus on the clustering task. We apply the deep learning method to the PPMI matrix instead of the Laplacian matrix used in their model. The former has been shown to have the potential to yield better representations than the latter (Perozzi, Al-Rfou, and Skiena 2014). To enhance our model's robustness, we also employ the stacked denoising autoencoders for learning multiple layers of representations.

We call our proposed model DNGR (deep neural networks for graph representations). The learned representations can be regarded as input features that can be fed into other tasks, such as unsupervised clustering and supervised classification tasks. To validate the effectiveness of our model, we conducted experiments on utilizing the learned representations for a spectrum of different tasks, where real-world networks of different genres and topologies are considered. To additionally demonstrate the effectiveness of our model when simpler, yet more large-scaled practical graph structures are considered, we applied our algorithm on a very large language dataset and conducted experiments on the word similarity task. In all such tasks, our model outperforms other methods for learning graph representations, and our model is also trivially parallelizable. Our major contribution is twofold:

- Theoretically, we argue that the deep neural networks offer the advantage of being able to capture non-linear information conveyed by the graph, whereas such information can not be readily captured by many widely used conventional linear dimension reduction methods. In addition, we argue that our random surfing model can be used to replace widely used conventional sampling based approaches for gathering graph information.
- Empirically, we demonstrate that our novel model is able to learn better low-dimensional vertex representations of a weighted graph, where meaningful semantic, relational and structural information of the graph can be captured. We show that the resulting representations can be effectively used as features for different down-stream tasks.

2 Background and Related Work

In this section, we first present the random sampling of an unweighed graph proposed in DeepWalk to illustrate the feasibility of transforming vertex representations into linear representations. Next, we consider two word representation methods: skip-gram with negative sampling and matrix factorization based on a PPMI matrix. These methods can be viewed as linear approaches to learning word representations from linear structural data.

Notation Given an weighted graph $G = \langle V, E \rangle$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of vertices and $E = \{e_{i,j}\}$ is the set of edges between two vertices. In an unweighted graph, edge weights indicate if a relationship exists between two vertices and are therefore binary. In contrast, edge weights in weighted graphs are real numbers denoting the degree of relatedness between two vertices. Although edge weights in weighted graphs can be negative, we only consider non-negative weights in this paper. For notational convenience, we also use w and c to denote vertices throughout

this paper. We seek to obtain a matrix R representing the vertices of graph G by capturing deep structural information.

2.1 Random Walk in DeepWalk

DeepWalk provides one effective way, called *truncated random walk*, to transform unweighted graph structural information into linear sequences, which express the relationship between vertices of a graph. The proposed random walk is a uniform sampling method that is suitable for unweighted graphs. They first select one vertex v_1 from the graph at random, and mark it as the current vertex, then randomly select the next vertex v_2 from all the neighbours of the current vertex v_1. Now, they mark this newly selected vertex v_2 as the current vertex and repeat such a vertex sampling process. When the number of vertices within one sequence reaches a pre-set number called walk length η, the algorithm terminates. After repeating the above procedure γ times (called total walk), chunks of linear sequences are gathered.

Although truncated random walk is one effective way to express unweighted graph structural information using linear sequences, the procedure involves a slow sampling process, and the hyper parameters η and γ are not easy to determine. We note that the DeepWalk yields a co-occurrence matrix based on the sampled linear sequences. In Section 3, we describe our random surfing model that directly construct a probabilistic co-occurrence matrix from a weighted graph, which avoids the expensive sampling process.

2.2 Skip-gram with Negative Sampling

Natural language corpora consist of streams of words which are linear sequences. Recently, neural embedding methods and matrix factorization based approaches have been widely used in learning word representations. The skip-gram model proposed in (Mikolov et al. 2013) has proven to be an effective and efficient way to learn word representations. Two notable ways to improve the skip-gram model are negative sampling (SGNS) and hierarchical softmax. In this paper, we choose to use the former approach.

In SGNS, a simplified variant of the *noise contrastive estimation* (NCE) approach (Gutmann and Hyvärinen 2012; Mnih and Teh 2012) is employed to enhance the robustness of the skip-gram model. SGNS stochastically creates negative pairs (w, c_N) from an empirical unigram word distributions and attempts to represent each word $w \in V$ and context word $c \in V$ pair using a low dimensional vector. The objective function of SGNS aims to maximize positive pairs (w, c) and minimize negative pairs (w, c_N).

$$\arg\max_{\vec{w},\vec{c}} \log \sigma(\vec{w} \cdot \vec{c}) + \lambda \cdot \mathbb{E}_{c_N \sim P_D}[\log \sigma(-\vec{w} \cdot \vec{c_N})]$$

where $\sigma(\cdot)$ is the sigmoid function defined as $\sigma(x) = (1 + e^{-x})^{-1}$ and λ is the number of negative samples. $\mathbb{E}_{c_N \sim p_D}[\cdot]$ is the expectation when the instance obtained from negative sampling c_N conforms to the distribution $p_D = \frac{\#(c)}{|D|}$ (the distribution of the unigram c in the dataset D).

2.3 PMI matrix and SVD

An alternative approach to learning graph representations (especially word representations) is based on matrix factorization techniques. Such approaches learn representations based on statistics of global co-occurrence counts, and can outperform neural network method based on separate local context windows in certain prediction tasks (Bullinaria and Levy 2012; Levy, Goldberg, and Dagan 2015).

An example of a matrix factorization method is *hyperspace analogue analysis* (Lund and Burgess 1996) which factorizes a word-word co-occurrence matrix to yield word representations. A major shortcoming of such an approach and related methods is that frequent words with relatively little semantic value such as stop words have a disproportionate effect on the word representations generated. Church and Hanks's pointwise mutual information (PMI) matrix (Church and Hanks 1990) was proposed to address this problem and has since been proven to provide better word representations (Bullinaria and Levy 2007):

$$PMI_{w,c} = \log\left(\frac{\#(w,c) \cdot |D|}{\#(w) \cdot \#(c)}\right)$$

where $|D| = \sum_w \sum_c \#(w, c)$.

A common approach to improving performance is to assign each negative value to 0, detailed in (Levy and Goldberg 2014), to form the PPMI matrix

$$X_{w,c} = max(PMI_{w,c}, 0)$$

where X is the PPMI matrix.

Although the PPMI matrix is a high dimensional matrix, performing dimension reduction using truncated SVD method (Eckart and Young 1936) yields the optimal rank d factorization with respect to L_2 loss (Levy and Goldberg 2014). We assume matrix X can be decomposed into three matrices $U\Sigma V^T$, where U and V are orthonormal matrices, and Σ is a diagonal matrix. In other words,

$$X \approx X_d = U_d \Sigma_d V_d^T$$

Here U_d and V_d are the left d columns of U and V corresponding to the top-d singular values (in Σ_d). According to Levy et al. (2015), the word representation matrix R can be:

$$R = U_d(\Sigma_d)^{1/2} \qquad \text{or} \qquad R = U_d$$

The PPMI matrix X is the product of the word representation matrix and the context matrix. The SVD procedure provides us a way of finding the matrix R from the matrix X, where the row vectors of R, which are the low-dimensional representations for words/vertices, can be obtained through a linear projection from their high-dimensional representations as given by X. We argue that such a projection does not have to be a linear projection. In this work, we investigate the use of non-linear projection methods to replace linear SVD approach through the use of deep neural networks.

2.4 Deep Neural Networks

Deep neural networks, which can be used to learn multiple levels of feature representations, has achieved successful results in different fields (Dahl et al. 2012; Krizhevsky, Sutskever, and Hinton 2012). Training such networks were shown difficult. One effective solution, proposed in (Hinton and Salakhutdinov 2006; Bengio et al. 2007), is to use the greedy layer-wise unsupervised pre-training. This strategy aims at learning useful representations of each layer at a time. The learned low-level representations are then fed

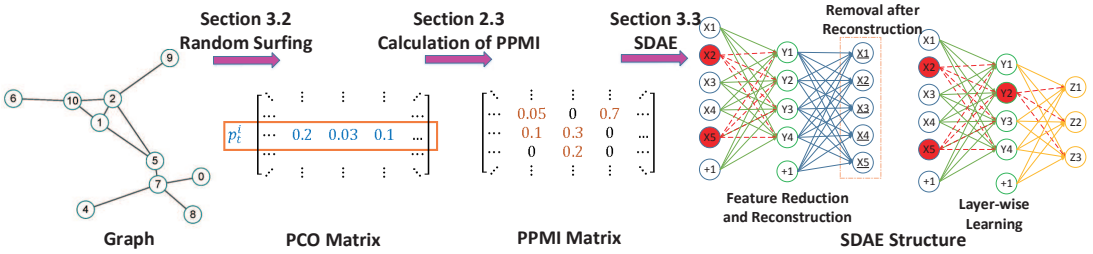


Figure 1: Main components: random surfing, calculation of PPMI matrix and feature reduction by SDAE

as the inputs to the next layer for subsequent representation learning. Neural networks typically employ non-linear activation function such as *sigmoid* or *tanh* to capture complex, non-linear projections from the inputs to the outputs.

To train a deep architecture that involves multiple layers of feature representations, *autoencoders* have emerged as one of the commonly used building blocks (Bourlard and Kamp 1988; Hinton and Zemel 1994). An autoencoder performs two actions – an encoding step, followed by a decoding step. In the encoding step, a function $f_{\theta_1}(\cdot)$ is applied to the vector in the input space and send it to a new feature space. An activation function is typically involved in this process to model the non-linearities between the two vector spaces – the space of input vectors and the space of latent vector representations. At the decoding step, a reconstruction function $g_{\theta_2}(\cdot)$ is used to reconstruct the original input vectors back from the latent representation space. Let us assume that $f_{\theta_1}(x) = \sigma(W_1 x + b_1)$ and $g_{\theta_2}(y) = \sigma(W_2 y + b_2)$, where $\sigma(\cdot)$ is the activation function, $\theta_1 = \{W_1, b_1\}$ are the weights (parameters) involved in the encoder, and $\theta_2 = \{W_2, b_2\}$ are the weights (parameters) involved in the decoder. Here W_1 and W_2 are linear maps (matrices) transforming the vectors from the input space, and b_1 and b_2 are the bias vectors. Our goal is to minimize the following reconstruction loss function by finding θ_1 and θ_2:

$$\sum_i L\big(x^{(i)}, g_{\theta_2}(f_{\theta_1}(x^{(i)}))\big)$$

where L is sample-wise loss, and $x^{(i)}$ is the i-th instance.

A stacked autoencoder is a multi-layer deep neural network consisting of multiple layers of such autoencoders. Stacked autoencoders use the layer-wise training approach to extract essential regularities capturing different levels of abstractions from the data layer by layer, with higher layers conveying higher-level abstractions from the data.

3 DNGR Model

We now detail our DNGR model. As shown in Figure 1, the model consists of three major steps. First, we introduce random surfing model (Sec 3.1) to capture graph structural information and generate a probabilistic co-occurrence matrix. Next we calculate the PPMI matrix based on our probabilistic co-occurrence matrix by following (Bullinaria and Levy 2007) (See Sec 2.3). After that, a stacked denoising autoencoder (Sec 3.2) is used to learn low-dimensional vertex representations.

3.1 Random Surfing and Context Weighting

Although effective, the sampling method for transforming a graph structure into linear sequences has some weaknesses. First, the sampled sequences have finite lengths. This makes it difficult to capture the correct contextual information for vertices that appear at the boundaries of the sampled sequences. Second, it is not straightforward to determine certain hyper parameters such as walk length η and total walks γ, especially for large graphs.

To resolve these issues, we consider using a random surfing model motivated by the PageRank model used for ranking tasks. We first randomly order the vertices in a graph. We assume our current vertex is the i-th vertex, and there is a transition matrix A that captures the transition probabilities between different vertices.

We introduce a row vector p_k, whose j-th entry indicates the probability of reaching the j-th vertex after k steps of transitions, and p_0 is the initial 1-hot vector with the value of the i-th entry being 1 and all other entries being 0. We consider a random surfing model with restart: at each time, there is a probability α that the random surfing procedure will continue, and a probability $1-\alpha$ that it will return to the original vertex and restart the procedure. This leads to the following recurrence relation:

$$p_k = \alpha \cdot p_{k-1} A + (1-\alpha) p_0$$

If we assume there is no random restart in the procedure, the probabilities for arriving at different vertices after exactly k steps of transitions is specified by the following:

$$p_k^* = p_{k-1}^* A = p_0 A^k$$

Intuitively, the distance between two vertices is closer, the more intimate relationship they should have. Thus, it is reasonable to weigh the importance of contextual nodes based on their relative distance to the current node. Such weighting strategies were implemented in both word2vec (Mikolov et al. 2013) and GloVe (Pennington, Socher, and Manning 2014) and were found important for achieving good empirical results (Levy, Goldberg, and Dagan 2015). Based on this fact, we can see that ideally the representation for the i-th vertex should be constructed in the following way:

$$r = \sum_{k=1}^{K} w(k) \cdot p_k^*$$

where $w(\cdot)$ is a decreasing function, *i.e.*, $w(t+1) < w(t)$.

We argue that the following way of constructing the vertex representation based on the above-mentioned random surfing procedure actually satisfies the above condition:

$$r = \sum_{k=1}^{K} p_k$$

In fact, it can be shown that we have the following:

$$p_k = \alpha^k p_k^* + \sum_{t=1}^{k} \alpha^{k-t}(1-\alpha) p_{k-t}^*$$

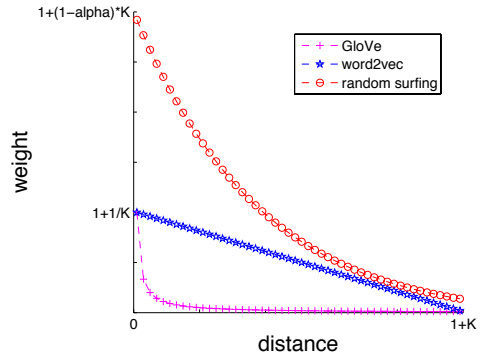


Figure 2: Different context weighting strategies in word2vec, Glove and random surfing. The weight assigned to a context node is a function of the distance between the context node and the current node in the graph.

where $p_0^* = p_0$. The coefficient of p_t^* in r is

$$w(t) = \alpha^t + \alpha^t(1-\alpha)(K-t)$$

where $w(t)$ is a decreasing function since $\alpha < 1$. While the weighting function in skip-gram can be described as

$$w^{'}(t) = -\frac{t}{K} + (1 + \frac{1}{K})$$

and the weighting function in GloVe is:

$$w^{''}(t) = \frac{1}{t}$$

As we can see from Figure 2, all the weighting functions are monotonically decreasing functions. Similar to word2vec and GloVe, our model also allows contextual information to be weighed differently based on their distance to the target, which was previously shown to be important for constructing good word representations. We will assess the importance of this aspect through experiments.

3.2 Stacked Denoising Autoencoder

Our study aims to investigate the construction of high-quality low-dimensional vector representations for vertices from the PPMI matrix that conveys essential structural information of the graph. The SVD procedure, though effective, essentially only yields *linear* transformations that map from vector representations contained by the PPMI matrix to the final low-dimensional vertex vector representations. We believe that potentially a *non-linear* mapping can be established between the two vector spaces. Thus we use stacked denosing autoencoders, a popular model used in deep learning, to generate compressed, low-dimensional vectors from the original high-dimensional vertex vectors.

As described in Table 1, we first initialize the parameters of the neural networks, and then we use the greedy layer-wise training strategy to learn high-level abstractions at each layer. In order to enhance the robustness of DNN, we use *stacked denoising autoencoders* (SDAE). Different from conventional autoencoders, denoising autoencoders partially corrupt the input data before taking the training step. Specifically, we corrupt each input sample x (a vector) randomly by assigning some of the entries in the vector to 0 with a certain probability. This idea is analogous to that of modeling missing entries in matrix completion tasks, where the goal is to exploit regularities in the data matrix to effectively recover the complete matrix under certain assumptions. In the

Table 1: SDAE description

Input: PPMI matrix X, Number of SDAE layers Γ
1. Initialize SDAE
Set number of nodes n_j in layer j, $X^{(1)} = X$
2. Greedy layer-wised training
For $j = 2$ to Γ
2.1 Construct one hidden layer SDAE with input of $X^{(j)}$
2.2 Learn hidden layer representation $h^{(j)}$
2.3 $X^{(j)} = h^{(j)}$ $\left(X^{(j)} \in \mathbb{R}^{n \times n_j}\right)$
Output: Matrix of vertices representations R

SDAE structure as shown in Figure 1, X_i's correspond to the input data, Y_i's correspond to the learned representations in the first layer, and Z_i's correspond to the learned representations in the second layer. The temporarily corrupted nodes (*e.g.* X_2, X_5 and Y_2) are highlighted with red color.

Similar to standard autoencoders, we again reconstruct data from the latent representations. In other words, we are interested in:

$$\min_{\theta_1,\theta_2} \sum_{i=1}^{n} L\big(x^{(i)}, g_{\theta_2}(f_{\theta_1}(\tilde{x}^{(i)}))\big)$$

where $\tilde{x}^{(i)}$ is corrupted input data of $x^{(i)}$, and L is the standard squared loss.

3.3 Discussions of the DNGR Model

Here, we analyze the advantages of DNGR. Its empirical effectiveness will be presented in Sec 4.

- **Random Surfing Strategy:** As we have analyzed above in Sec 3.1, the random surfing procedure overcomes the limitations associated with the sampling procedures used in previous works, and it provides us with certain desired properties which are shared by previous implementations to embedding models, such as word2vec and GloVe.
- **Stacking Strategy:** The stacked structure provides a smooth way of performing dimension reduction. We believe the learned representations at different layers of the deep architecture can provide different levels of meaningful abstractions to the input data.
- **Denoising Strategy:** High dimensional input data often contains redundant information and noise. It is believed that the denoising strategy can effectively reduce noise and enhance robustness (Vincent et al. 2008).
- **Efficiency:** Following previous analysis (Tian et al. 2014), inference with the autoencoder used in DNGR has a lower time complexity than SVD-based matrix factorization method with respect to the number of vertices in the graph. SVD has a time complexity at least quadratic in the number of vertices. However, the inference time complexity of DNGR is linear in the number of vertices in the graph.

4 Datasets, Baselines and Experiments

To assess the performance of the DNGR model, we conducted experiments on real datasets. Besides showing results for graph representations, we also assess the effectiveness of word embeddings learned from a natural language corpus consisting of linear sequences.

Table 2: Neural Network Structures

dataset	#nodes in each layer
Wine	[178-128-32]
3NG	[600-512-256]
6NG	[1200-512-256]
9NG	[1800-1024-512-256]
Word Representation	[10000-4096-2048-1024-256]

4.1 Datasets

1. **20-NewsGroup** contains around $20,000$ News Group (NG) documents and is split into 20 different groups according to category. In our experiments, each document is a single vertex represented by a word vector produced via TF-IDF and the weight of the edges between two documents is the cosine similarity of the two documents. To show the robustness of our model, as demonstrated in (Tian et al. 2014), we constructed 3 groups from 3, 6 and 9 different categories respectively, and randomly selected 200 documents from each category. In our experiments, we selected the same categories as those used in (Tian et al. 2014). These three networks are represented by fully-connected, weighted graphs which contain 200, 400 and 600 documents respectively.

2. **Wine**, a dataset from UCI Machine Learning Repository (Lichman 2013), is the the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the 3 types of wines and comprises 178 instances. Following (Tian et al. 2014), we built a graph using cosine similarity of two different instances as edge weights, and labels as ground truth.

3. **Wikipedia Corpus** is a free and open content online encyclopedia. We picked the April 2010 snapshot from (Shaoul 2010) used in (Huang et al. 2012) as our training corpus for learning word representations. The dataset has 2 million articles and 990 million tokens. To demonstrate the effectiveness of deep learning as compared to SVD, we chose a dictionary of 10,000 most frequent words (10,000 selected due to the time complexity of the SVD algorithm).

To evaluate the performance of word representations generated by each algorithm, we conducted experiments on word similarities (Mikolov et al. 2013) on 4 datasets, including the popular WordSim353 (Finkelstein et al. 2001), WordSim Similarity (Zesch, Müller, and Gurevych 2008) and WordSim Relatedness (Agirre et al. 2009) and MC (Miller and Charles 1991), as used in (Pennington, Socher, and Manning 2014; Levy, Goldberg, and Dagan 2015). All datasets have word pairs and human-assigned similarities.

4.2 Baseline Algorithms

We consider the following four baseline algorithms[1].

1. **DeepWalk** (Perozzi, Al-Rfou, and Skiena 2014) is a recently proposed method to learn representations of networks. It transforms a graph structure into linear sequences by truncated random walks and processes the sequences using skip-gram with hierarchical softmax.

[1] Since we are considering weighted graphs in this work, for the first two baseline algorithms, we used *weighted sampling* to sample sequences based on the weights on the edges of the graph rather than adopting the *uniform sampling* method used in their original papers which is only applicable to unweighted graphs.

Table 3: Clustering Results on 20-NewsGroup

NMI (%)	3NG	6NG	9NG
DNGR(α=0.98)	**80.44**	68.76	**59.13**
DNGR(α=0.95)	74.30	**68.87**	57.32
DNGR(α=1)	74.63	66.70	58.22
SVD (α=0.98)	62.95	66.18	52.90
SVD (α=0.95)	62.83	65.33	50.00
SVD (α=1)	60.27	63.52	53.52
PPMI	62.84	66.52	52.30
DeepWalk (128)	55.83	58.11	46.86
DeepWalk (256)	56.34	58.40	46.62
DeepWalk (512)	59.64	59.82	46.29
SGNS (128)	64.34	62.99	48.14
SGNS (256)	62.56	63.25	48.10
SGNS (512)	63.97	63.16	49.04

2. **SGNS** (Mikolov et al. 2013) is proposed in word2vec. It is suitable for capturing linear structures. SGNS has been proven to be an effective model to learn word representations both theoretically and empircally.
3. **PPMI** (Bullinaria and Levy 2007) is a measure often used in information theory. PPMI was used for word representations in (Levy, Goldberg, and Dagan 2015) and is a sparse high dimensional representation.
4. **SVD** (Levy and Goldberg 2014) is a common matrix factorization method that is used to reduce dimensions or extract features. Following (Levy and Goldberg 2014), we used SVD to compress the PPMI matrix to obtain low dimensional representations.

5.1 Parameters

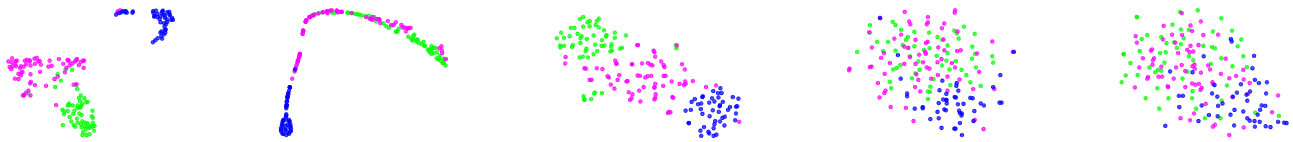
In baseline systems, we set walk length $\eta = 40$, total walks $\gamma = 80 * |V|$, where $|V|$ is the number of graph vertices as suggested in (Perozzi, Al-Rfou, and Skiena 2014). To reach optimal performance for each baseline algorithm, we set negative samples $\lambda = 5$, context window size $K = 10$ for DeepWalk and SGNS, as stated in (Levy, Goldberg, and Dagan 2015). For our model, we tuned the *dropout ratio* which reduces error in the optimization process. As listed in Table 2, the neural networks for 3NG and 6NG have 3 layers and 9NG and Blogcatalog have 4 layers. All neurons were activated by the sigmoid function.

5.2 Experiments

We present empirical results of our proposed method against each baseline algorithm. The results demonstrate the advantage of using the weighting strategy by our random surfing procedure and the effectiveness of deep architectures. Our source code will be released at http://shelson.top.

5.2.1 Clustering Task on the 20-NewsGroup Network

In this experiment, we ran each baseline algorithm to generate a representation vector for each vertex, which is used as a feature representation for clustering. We used the K-means algorithm (Arthur and Vassilvitskii 2007), and the *normalized mutual information* (NMI) (Strehl, Ghosh, and Mooney 2000) as the metric to measure performance. We ran each algorithm 10 times with different initializations and reported the average scores in Table 3. To illustrate the impact of different dimensions, we also listed the results of 128 and 512 dimensions for DeepWalk and SGNS.



(a) DNGR (α=0.98) (b) DNGR(α=1) (c) SVD (α=0.98) (d) DeepWalk (e) SGNS

Figure 3: Visualization Results of Wine. Vector representations serve as features to feed into t-SNE tool and points in the same color indicate instances of the same type.

From the results, we can see that our DNGR model significantly outperform other baseline algorithms, especially when α=0.98. When α=1, contextual information is not weighed based on their distance, and we obtain lower results, confirming the importance of the weighing strategy discussed in Sec 3.1. Increasing the dimensions used in DeepWalk and SGNS did not lead to consistent improvements in scores. Comparing DNGR and PPMI, the improvement in scores demonstrated the benefits of extracting meaningful information and reducing dimensions for sparse high dimensional matrices. Under the same setting of α, DNGR consistently gives better results than SVD. This provides empirical justifications to our previous discussions – the fact that SVD does not always outperform PPMI indicates that potentially a better dimension reduction method is needed.

5.2.2 Visualization of Wine Another way of assessing the quality of the vertex representations is through visualization. We conducted visualization experiments by following (Tang et al. 2015) to compare the performance of DNGR with SVD, deepwalk and SGNS on Wine. The vertex representation matrix was fed as features into t-SNE (Van der Maaten and Hinton 2008), which mapped all points into a 2D space, where the same type of wines were highlighted with the same color. Under the same setting, a clustering with clearer boundaries between different color groups indicates better representations.

From Figure 3, we can see that the visualization of outputs from DeepWalk and SGNS showed unclear boundaries and diffuse clusters. DNGR(α=1) is much better. Although SVD is superior to DNGR(α=1), in the results we can see the green points are still intermixed with some red points. DNGR(α=0.98) is the clear winner. We can see that points of different colors appear in 3 separate poles and the majority of the points of the same color are clustered together.

5.2.3 Word Similarity Task To further understand the effectiveness of the deep learning component of our DNGR model, we conducted experiments on a large scale language corpus, where we learn word representations from large linear structured graph data. In this experiment, since the data consists of strictly linear sequences, the random surfing procedure is not required. Instead, we could directly count word-word pairs from training corpus. To compare the performance of the different algorithms, we conducted evaluations on a word similarity task over standard datasets. Spearman's rank correlation coefficient (Zar 1972) was used to evaluate the results produced by the algorithms. Following the findings reported in (Levy, Goldberg, and Dagan 2015), to obtain optimal results, we set the number of negative samples to 5, 1, 1 for SGNS, DNGR and SVD respectively.

Table 4: Results of Word Similarity Task

Algorithm	Finkelstein et al. WS353	Zesch et al. WS Similarity	Agirre et al. WS Relatedness	Miller et al. MC
DNGR	**72.45**	**74.84**	**68.73**	**84.64**
PPMI	62.28	67.52	56.02	66.07
SVD	72.22	73.96	68.44	83.57
SGNS	70.78	74.37	64.60	75.36

Table 5: Layer-wise NMI Values on 20-NewsGroup

Layer	1	2	3	4
3NG	62.84	75.83	80.44	—
6NG	66.25	66.46	68.76	—
9NG	52.30	56.09	56.52	59.13

As shown in Table 4, the performance of DNGR outstrips other baseline algorithms. Both SVD and DNGR yields significantly better results than PPMI, demonstrating the importance of the dimension reduction in this task. The performance of DNGR is higher than SVD and SGNS across all datasets, which reveals the importance of capturing complex non-linear regularities when learning representations, and illustrates the capability of learning better abstractions using our deep learning approach.

5.2.4 Importance of Deep Structures To understand necessity of deep structures, we further measured the layer-wise NMI value of 20-NewsGroup. For 3NG and 6NG, we use a 3-layer neural network to construct representations, and for 9NG we used 4 layers as shown in Table 2. From Table 5, we can see that the performance in general increased from shallow layers to deep layers for all three networks, which demonstrated the importance of deep structures.

6 Conclusion

In this paper, we have proposed a deep graph representation model for encoding each vertex as a low dimensional vector representation. Our model demonstrated the ability of stacked denosing autoencoders in extracting meaningful information and generating informative representations. We conducted experiments on real-world datasets in different tasks and showed that the performance of the proposed model outperformed several state-of-the-art baseline algorithms. Although the process of tuning the hyper-parameters of neural networks requires efforts and expertise, we conclude that exploring the underlying structure of data via deep learning can lead to improved representations for graphs.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments, and Si Kai Lee for his help with this work. This work was done when the first author was visiting SUTD. This work was supported by SUTD grant ISTD 2013 064 and Temasek Lab project IGDST1403013.

References

Agirre, E.; Alfonseca, E.; Hall, K.; Kravalova, J.; Paşca, M.; and Soroa, A. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *NAACL*, 19–27.

Arthur, D., and Vassilvitskii, S. 2007. k-means++: The advantages of careful seeding. In *SODA*, 1027–1035.

Bengio, Y.; Lamblin, P.; Popovici, D.; Larochelle, H.; et al. 2007. Greedy layer-wise training of deep networks. In *NIPS*, 153–160.

Bourlard, H., and Kamp, Y. 1988. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics* 59(4-5):291–294.

Bullinaria, J. A., and Levy, J. P. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior research methods* 39(3):510–526.

Bullinaria, J. A., and Levy, J. P. 2012. Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and svd. *Behavior research methods* 44(3):890–907.

Cao, S.; Lu, W.; and Xu, Q. 2015. Grarep: Learning graph representations with global structural information. In *CIKM*, 891–900.

Church, K. W., and Hanks, P. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics* 16(1):22–29.

Dahl, G. E.; Yu, D.; Deng, L.; and Acero, A. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on* 20(1):30–42.

Eckart, C., and Young, G. 1936. The approximation of one matrix by another of lower rank. *Psychometrika* 1(3):211–218.

Finkelstein, L.; Gabrilovich, E.; Matias, Y.; Rivlin, E.; Solan, Z.; Wolfman, G.; and Ruppin, E. 2001. Placing search in context: The concept revisited. In *WWW*, 406–414.

Gutmann, M. U., and Hyvärinen, A. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *JMLR* 13(1):307–361.

Hinton, G. E., and Salakhutdinov, R. R. 2006. Reducing the dimensionality of data with neural networks. *Science* 313(5786):504–507.

Hinton, G. E., and Zemel, R. S. 1994. Autoencoders, minimum description length, and helmholtz free energy. In *NIPS*, 3–10.

Huang, E. H.; Socher, R.; Manning, C. D.; and Ng, A. Y. 2012. Improving word representations via global context and multiple word prototypes. In *ACL*, 873–882.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*, 1097–1105.

Levy, O., and Goldberg, Y. 2014. Neural word embedding as implicit matrix factorization. In *NIPS*, 2177–2185.

Levy, O.; Goldberg, Y.; and Dagan, I. 2015. Improving distributional similarity with lessons learned from word embeddings. *TACL* 3:211–225.

Lichman, M. 2013. UCI machine learning repository.

Lund, K., and Burgess, C. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers* 28(2):203–208.

Macskassy, S. A., and Provost, F. 2003. A simple relational classifier. Technical report, DTIC Document.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, 3111–3119.

Miller, G. A., and Charles, W. G. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes* 6(1):1–28.

Mnih, A., and Teh, Y. W. 2012. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *EMNLP*, 1532–1543.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*, 701–710.

Shaoul, C. 2010. The westbury lab wikipedia corpus. *Edmonton, AB: University of Alberta*.

Strehl, A.; Ghosh, J.; and Mooney, R. 2000. Impact of similarity measures on web-page clustering. In *AAAI*, 58–64.

Tang, L., and Liu, H. 2009a. Relational learning via latent social dimensions. In *SIGKDD*, 817–826.

Tang, L., and Liu, H. 2009b. Scalable learning of collective behavior based on sparse social dimensions. In *CIKM*, 1107–1116.

Tang, L., and Liu, H. 2011. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery* 23(3):447–478.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *WWW*, 1067–1077.

Tian, F.; Gao, B.; Cui, Q.; Chen, E.; and Liu, T.-Y. 2014. Learning deep representations for graph clustering. In *AAAI*.

Van der Maaten, L., and Hinton, G. 2008. Visualizing data using t-sne. *JMLR* 9(2579-2605):85.

Vincent, P.; Larochelle, H.; Bengio, Y.; and Manzagol, P.-A. 2008. Extracting and composing robust features with denoising autoencoders. In *ICML*, 1096–1103.

Zar, J. H. 1972. Significance testing of the spearman rank correlation coefficient. *Journal of the American Statistical Association* 67(339):578–580.

Zesch, T.; Müller, C.; and Gurevych, I. 2008. Using wiktionary for computing semantic relatedness. In *AAAI*, volume 8, 861–866.