Tri-Party Deep Network Representation

Shirui Pan[†], Jia Wu[†], Xingquan Zhu[*], Chengqi Zhang[†], Yang Wang[‡]
[†]Centre for Quantum Computation & Intelligent System, FEIT, University of Technology Sydney
[*] Dept. of Computer & Electrical Engineering and Computer Science, Florida Atlantic University, USA
[‡]The University of New South Wales, Australia
{shirui.pan, jia.wu, chengqi.zhang}@uts.edu.au; xzhu3@fau.edu; wangy@cse.unsw.edu.au

Abstract

Information network mining often requires examination of linkage relationships between nodes for analysis. Recently, network representation has emerged to represent each node in a vector format, embedding network structure, so off-the-shelf machine learning methods can be directly applied for analysis. To date, existing methods only focus on one aspect of node information and cannot leverage node labels. In this paper, we propose TriDNR, a tri-party deep network representation model, using information from three parties: node structure, node content, and node labels (if available) to jointly learn optimal node representation. TriDNR is based on our new coupled deep natural language module, whose learning is enforced at three levels: (1) at the network structure level, TriDNR exploits *inter-node relationship* by maximizing the probability of observing surrounding nodes given a node in random walks; (2) at the node content level, TriDNR captures *node-word correlation* by maximizing the co-occurrence of word sequence given a node; and (3) at the node label level, TriDNR models *label-word correspondence* by maximizing the probability of word sequence given a class label. The tri-party information is jointly fed into the neural network model to mutually enhance each other to learn optimal representation, and results in up to 79% classification accuracy gain, compared to state-of-the-art methods.

1 Introduction

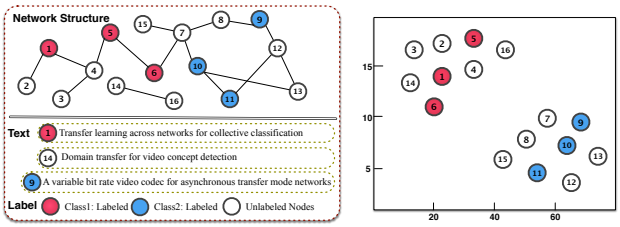
Many network applications, such as social networks, protein networks, and citation networks, are characterized with complex structure and rich node content information, where the network structure is naturally sparse in that only a small number of nodes are connected and the node content is usually represented as text information indicating the properties of a node, such as the title or abstract information of each paper (node) in a citation network. The complexity of networked data imposes great challenge to many machine learning tasks such as node classification in networks. To address this problem, network representation [Perozzi *et al.*, 2014; Tang *et al.*, 2015; Yang *et al.*, 2015; Tian *et al.*, 2014; Chang *et al.*, 2015] encodes each node in a common, continuous, and low-dimensional space while preserving the neighborhood relationship between nodes, so machine learning algorithms can be applied directly. For network representation, existing methods mainly employ network structure based methods or node content based methods.

Majority network representation methods are based on network structure. Early works aim to learn social dimensions [Tang and Liu, 2009] or knowledge bases [Bordes *et al.*, 2011; Socher *et al.*, 2013] to embed the entities in a network. Inspired by the deep learning techniques in natural language processing [Mikolov *et al.*, 2013], Perozzi *et al.* [Perozzi *et al.*, 2014] recently proposed a DeepWalk algorithm which employs neural network models to learn a feature vector for each node from a corpus of random walks generated from networks. These methods take the network structure as input but ignore content information associated to each node.

From the content perspective, many methods exist to represent a text message into a vector space. Early studies employ bag of word approaches (*e.g.*, TFIDF) or topic models (*e.g.* LDA [Blei *et al.*, 2003]) to represent each document as a vector. However, these models do not consider the context information of a document (*i.e.*, order of words), resulting in suboptimal representation. Recently, neural network or deep learning based approaches [Mikolov *et al.*, 2013; Le and Mikolov, 2014; Luong *et al.*, 2013] have emerged for text embedding. The Skip-gram model [Mikolov *et al.*, 2013] employs a simple neural network model to learn distributed vectors for words. Due to its simplicity, efficiency, and scalability, Skip-gram model is further extended as paragraph vector model [Le and Mikolov, 2014], which learns latent vector representation for arbitrary piece of text.

The drawback of existing methods [Tang and Liu, 2009; Perozzi *et al.*, 2014; Tang *et al.*, 2015; Le and Mikolov, 2014], regardless of network exploration approaches or text modeling methods, is twofold: (1) they only utilize one source of information, so the representation is inherently *shallow*. (2) all these methods learn network embedding in a fully unsupervised way. For many tasks, such as node classification, we may have a limited number of labeled nodes, which provides usefully information to assist network representation.

When considering networks as a whole, the main challenge of learning latent representation for network nodes is twofold:



(a) Input: A Network (b) Output: Representation

Figure 1: Our model takes an information network as input and outputs a low-dimensional representation for each node. In this toy citation network, each node denotes a paper, links are citation relationships, and texts are paper titles. Red and blue nodes are labeled nodes from two classes (transfer learning *vs.* video coding). Remaining nodes are unlabeled. (1) paper 1 and paper 9 share two common words *transfer* and *networks* in their titles. If we ignore their labels, they will likely be represented with similar representation. So label is useful. (2) paper 14 and paper 1 belong to the *transfer learning* class. Although paper 14 is more similar to paper 9 in terms of shared title words, the weak citations/connections between paper 1 and paper 14 will make them close to each in the learned node representation space. So structure is useful for node representation. (3) By combining structure, text, and labels, our method is positioned to learn best representations.

- **Network Structure, Node Content, and Label Information Integration (Challenge 1)** How to learn node embedding for networks containing links and rich text information, and further tailor the representation for learning tasks provided with labeled samples?
- **Neural Network Modeling (Challenge 2)** How to design effective neural network models for networked data, in order to obtain deep network representation results?

In terms of neural network modeling (Challenge 2), a straightforward method for this problem is to separately learn a node vector by using DeepWalk [Perozzi *et al.*, 2014] for network structure and learn a document vector via Paragraph Vectors model [Le and Mikolov, 2014] (the state-of-the art model to embed text into a vector space), and then concatenate these two vectors into a unified representation. However this simple combination is suboptimal because it ignores the label information, and overlooks interactions between network structures and text information.

In order to exploit both structure and text information for network representation (Challenge 1), a recent work TADW [Yang *et al.*, 2015] shows that DeepWalk is equivalent to factorize a matrix M (sum of a series transition matrix). However, this method has following drawbacks: (1) The accurate matrix M for factorization is non-trivial and very difficult to obtain. As a result, TADW has to factorize an approximate matrix, which will weaken its representation power; (2) TADW simply ignores the context of text information (*e.g.*, orders of words), so cannot appropriately capture the semantics of the words and nodes in a network setting; (3) unlike neural network models that can scale up to millions of records in a single machine [Mikolov *et al.*, 2013], TADW requires expensive matrix operation like SVD decomposition, which prohibits TADW from handling large scale data.

In this paper, we propose a tri-party deep network representation model, TriDNR, which uses a coupled neural network architecture to exploit network information from three parties: node structure, node content, and node labels (if available). At the node level, we maximize the probability of observing the surrounding nodes given a node v_i in a random walk sequence, which well preserves *inter-node relationship* in networks. At the node content and node label levels, we maximize the co-occurrence of word sequence given a node and co-occurrence of word sequence given a label, which accurately captures the *node-word correlation* and *label-word correspondence*. The tri-party information mutually enhances each other for optimal node representation. An example of using TriDNR to represent nodes of a citation network to a two dimensional space is shown in Fig 1.

2 Problem Definition

An information network is represented as $G = (V, E, D, \mathcal{C})$, where $V = \{v_i\}_{i=1,\cdots,N}$ consists of a set of nodes, $e_{i,j} = (v_i, v_j) \in E$ is an edge encoding the edge relationship between the nodes, $d_i \in D$ is a text document associated with each node v_i, $\mathcal{C} = L \bigcup U$ is the class label information of the network, with L denoting the labeled nodes and U being the unlabeled nodes. The network representation **aims** to learn a low-dimensional vector $\mathbf{v}_{v_i} \in \mathbb{R}^k$ (k is a smaller number) for each node v_i in the network, so that nodes close to each other in network topology or with similar text content, or sharing the same class label information are close in the representation space. Fig 1 demonstrates the importance of combining structure, text, and labels for learning good representations.

In this paper, we assume the network has partial labeled nodes. If the label set $L = \emptyset$, the representation becomes pure unsupervised, and our proposed solution is still valid.

3 Preliminary: Skip-Gram and DeepWalk

Recently, Skip-Gram [Mikolov *et al.*, 2013], which is a language model exploiting word orders in a sequence and assuming that words closer are statistically more dependent or related, has drawn much attention due to its simplicity and efficiency. Specifically, Skip-Gram aims to predict the *context* (surrounding words) within a certain window given current word by maximizing the following log-likelihood:

$$\mathcal{L} = \sum_{t=1}^{T} \log \mathbb{P}(w_{t-b} : w_{t+b} | w_t) \tag{1}$$

where b is the context width (window size), $w_{t-b} : w_{t+b}$ is a sequence of words excluding the word w_t itself. The probability $\mathbb{P}(w_{t-b} : w_{t+b} | w_t)$ is computed as:

$$\prod_{-b \le j \le b, j \neq 0} \mathbb{P}(w_{t+j} | w_t) \tag{2}$$

Eq. (2) assumes that the contextual words $w_{t-b} : w_{t+b}$ are independent given word w_t. $\mathbb{P}(w_{t+j} | w_t)$ is computed as:

$$\mathbb{P}(w_{t+j} | w_t) = \frac{\exp(\mathbf{v}_{w_t}^\top \mathbf{v}'_{w_{t+j}})}{\sum_{w=1}^{W} \exp(\mathbf{v}_{w_t}^\top \mathbf{v}'_w)} \tag{3}$$

where $\mathbf{v}_w$ and $\mathbf{v}'_w$ are the input and output vectors of w. After training the model, the input vector $\mathbf{v}_w$ can be used as the final representation of word w.

3.1 DeepWalk Model

Motivated by the Skip-Gram, DeepWalk [Perozzi *et al.*, 2014] constructs a corpus S that consists of random walks generated from the network. Each random walk $v_1 \rightarrow v_3 \rightarrow \cdots \rightarrow v_n$ can be considered as a sentence and each node v_i can be considered as a word in neural language models. Then DeepWalk algorithm trains the Skip-Gram model [Mikolov *et al.*, 2013] on random walks, and obtain a distributed representation vector for each node. The objective of DeepWalk model is to maximize the likelihood of the surrounding nodes given current node v_i for all random walks $s \in \mathcal{S}$:

$$\begin{aligned}\mathcal{L} &= \sum_{i=1}^{N}\sum_{s\in\mathcal{S}} \log \mathbb{P}(v_{i-b} : v_{i+b}|v_i) \\ &= \sum_{i=1}^{N}\sum_{s\in\mathcal{S}}\sum_{-b\le j\le b, j\neq 0} \log \mathbb{P}(v_{i+j}|v_i)\end{aligned} \tag{4}$$

where N is the total number of nodes. The architecture of DeepWalk model is illustrated in the left panel of Fig 2.

DeepWalk can only utilize the network information for model learning, without considering any text information augmented with each node and label information provided by the specific task like node classification.

4 Tri-Party Deep Network Representation

In this section, we present our tri-party deep network representation algorithm for jointly utilizing network structure, text content, and label information to learn a latent vector for each node in the network.

We use neural network models to learn $\mathbf{v}_{v_i}$, the latent representation for node v_i in a network. This latent representation $\mathbf{v}_{v_i}$ acts as an input vector in our neural network model. More specifically, our TriDNR algorithm consists of two steps:

1. **Random Walk Sequence Generation** uses network structure as the input and randomly generates a set of walks over the nodes, with each walk rooting at a node v_i and randomly jumping to other nodes each time. The random walk corpus can capture the node relationship.
2. **Coupled Neural Network Model Learning** embeds each node into a continuous space by considering the following information: (a) random walk corpus which captures the inter-node relationship, (b) the text corpus which models the node-content correlations, (c) label information which encodes label-node correspondences.

4.1 Model Architecture

Our coupled neural network model architecture is illustrated in the right panel of Fig 2, which has the following properties:

1. **Inter-node Relationship Modeling.** The upper layer of TriDNR learns the structure relationship from the random walk sequences, under assumption that connected nodes are statistically dependent on each other.

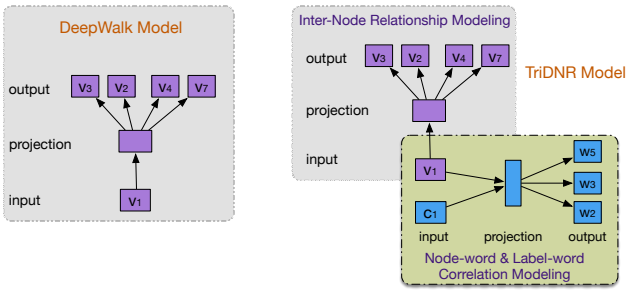


Figure 2: Architecture of the DeepWalk (Skip-Gram) method and our proposed TriDNR method. The DeepWalk approach learns the network representation based on the network structure only. Our TriDNR method couples two neural networks to learn the representation from three parties (*i.e.*, node structure, node content, and node label) to capture the *inter-node*, *node-word*, and *label-word* relationship. The *input*, *projection*, and *output* indicate the input layer, hidden layer, and output layer of a neural network model.

2. **Node-content correlations Assessing.** The lower layer of TriDNR models the contextual information (node-content correlations) of words within a document.

3. **Connections.** We couple these two layers by the node v_1 in the model, indicating that v_1 is influenced by both random walk sequences and node content information.

4. **Label-content Correspondence Modeling** To utilize the valuable class label information of each node, we also use the label of each document as input and simultaneously learn the input label vectors and output word vectors, providing directly modeling between node labels and node content.

Note that the label information is not used for inter-node relationship modeling. This is because we can hardly assess the class label of a random walk sequence.

The lower level (panel) of Fig 2, which exploits document and class label information, can be formalized by the following objective function:

$$\mathcal{L} = \sum_{i=1}^{|L|} \log \mathbb{P}(w_{-b} : w_b|c_i) + \sum_{i=1}^{N} \log \mathbb{P}(w_{-b} : w_b|v_i) \tag{5}$$

where $w_{-b} : w_b$ is a sequence of words inside a contextual window of length b. c_i is the class label of node v_i. Note that if no label information is used (*i.e.*, $|L| = 0$), the first term vanishes, Eq. (5) will become the Paragraph Vector model [Le and Mikolov, 2014], which exploits the text information to learn vector representation for each document.

Given a network G which consists of N nodes (*i.e.*, $V = \{v_i\}_{i=1,\cdots,N}$), suppose the random walks generated in G is $\mathcal{S}$, our TriDNR model aims to maximize the following log-

likelihood:

$$\mathcal{L} = (1-\alpha)\sum_{i=1}^{N}\sum_{s\in\mathcal{S}}\sum_{-b\le j\le b, j\neq 0}\log \mathbb{P}(v_{i+j}|v_i) + \alpha\sum_{i=1}^{N}\sum_{-b\le j\le b}\log \mathbb{P}(w_j|v_i) + \alpha\sum_{i=1}^{|L|}\sum_{-b\le j\le b}\log \mathbb{P}(w_j|c_i) \tag{6}$$

where α is the weight that balances network structure, text, and label information. b is the window size of sequence, and w_j indicates the j-th word in a contextual window.

Given equation Eq. (6), the first term requires the calculation of $\mathbb{P}(v_{i+j}|v_i)$, the probability of observing surrounding nodes given current node v_i, which can be computed using the soft-max function as follows:

$$\mathbb{P}(v_{i+j}|v_i) = \frac{\exp(\mathbf{v}_{v_i}^\top \mathbf{v}'_{v_{i+j}})}{\sum_{v=1}^{N}\exp(\mathbf{v}_{v_i}^\top \mathbf{v}'_v)} \tag{7}$$

where $\mathbf{v}_v$ and $\mathbf{v}'_v$ are the input and output vector representation of node v. Furthermore, the probability of observing contextual words $w_{-b} : w_b$ given current node v_i is:

$$\mathbb{P}(w_j|v_i) = \frac{\exp(\mathbf{v}_{v_i}^\top \mathbf{v}'_{w_j})}{\sum_{w=1}^{W}\exp(\mathbf{v}_{v_i}^\top \mathbf{v}'_w)} \tag{8}$$

where $\mathbf{v}'_{w_j}$ is the output representation of word w_j and W is the number of distinct words in the whole network. Similarly, the probability of observing the words given a class label c_i is defined as:

$$\mathbb{P}(w_j|c_i) = \frac{\exp(\mathbf{v}_{c_i}^\top \mathbf{v}'_{w_j})}{\sum_{w=1}^{W}\exp(\mathbf{v}_{c_i}^\top \mathbf{v}'_w)} \tag{9}$$

From Eq. (8) and Eq. (9), we know that the text information and label information will jointly affect $\mathbf{v}'_{w_j}$, the output representation of word w_j, which will further propagate back to influence the input representation of $v_i \in V$ in the network. As a result, the node representation (*i.e.*, the input vectors of nodes) will be enhanced by both network structure, text content, and label information.

4.2 Model Optimization

We train our model Eq. (6) using stochastic gradient ascent, which is suitable for large-scale data. However, computing the gradient in Eq. (7) is expensive, as it is proportional to the number of nodes in the network N. Similarly, computing the gradient in Eq. (8) and Eq. (9) are proportional to the unique number of words in the document content W. To handle this problem, we resort to the hierarchical soft-max [Morin and Bengio, 2005], which reduces the time complexity to $\mathcal{O}(R\log(W) + N\log(N))$ where R is the total number of words in the document content.

Specifically, the hierarchical model in our algorithm uses two binary trees, one with distinct nodes as leaves and another with distinct words as leaves. The tree is built using Huffman algorithm so that each vertex in the tree has a binary code and more frequent nodes (or words) has shorter codes. There is a unique path from the root to each leaf. The interval vertices of the trees are represented as real-valued vector with the same dimension as the leaves. So instead of enumerating all nodes in Eq. (7) in each gradient step, we only need to evaluate the path from the root to the corresponding leaf in the Huffman trees. Suppose the path to the leaf node v_i is a sequence of vertices $(l_0, l_1, \cdots, l_P)$, ($l_0 =$ root, $l_P = v_i$), then

$$\mathbb{P}(v_{i+j}|v_i) = \prod_{t=1}^{P}\mathbb{P}(l_t|v_i) \tag{10}$$

where $\mathbb{P}(l_t|v_i)$ can be further modeled by a binary classifier which is defined as:

$$\mathbb{P}(l_t|v_i) = \sigma(\mathbf{v}_{v_i}^\top \mathbf{v}'_{v_{l_t}}) \tag{11}$$

here $\sigma(x)$ is the sigmoid function, and $\mathbf{v}'_{v_{l_t}}$ is the representation of tree vertex l_t's parent. By doing this, the time complexity is reduced to $\mathcal{O}(N\log N)$. Similarly, we can use hierarchical soft-max technique to compute the words and labels in Eq. (8) and Eq. (9).

5 Experimental Results

5.1 Experimental Setup

We report our experimental results on two networks. Both of them are citation networks and we use the paper title as node content for each node in the networks.

DBLP dataset [1] consists of bibliography data in computer science [Tang *et al.*, 2008]. Each paper may cite or be cited by other papers, which naturally forms a citation network. In our experiments, we select a list of conferences from 4 research areas, *database* (SIGMOD, ICDE, VLDB, EDBT, PODS, ICDT, DASFAA, SSDBM, CIKM), *data mining* (KDD, ICDM, SDM, PKDD, PAKDD), *artificial intelligent* (IJCAI, AAAI, NIPS, ICML, ECML, ACML, IJCNN, UAI, ECAI,COLT, ACL, KR), *computer vision* (CVPR, ICCV, ECCV, ACCV, MM, ICPR, ICIP, ICME). The DBLP network consist of 60,744 papers (nodes), 52,890 edges in total.

CiteSeer-M10 is a subset [Lim and Buntine, 2014] of CiteSeer^X data [2] which consist of scientific publications from 10 distinct research areas: *agriculture*, *archaeology*, *biology*, *computer science*, *financial economics, industrial engineering, material science, petroleum chemistry, physics*, and *social science*. This dataset consists of multidisciplinary 10 classes, with 10,310 publications and 77,218 edges in total.

The following algorithms are comparing in our paper:

1. **DeepWalk** [Perozzi *et al.*, 2014] learns network representation using network structure only.
2. **LINE** [Tang *et al.*, 2015] is a state-of-the-art algorithm for network representation based on network structure.
3. **Doc2Vec** [Le and Mikolov, 2014] is the Paragraph Vectors algorithm which embeds any piece of text in a distributed vector using neural network models. Here we use PV-DBOW model in [Le and Mikolov, 2014].
4. **LDA** [Blei *et al.*, 2003] is the Latent Dirichlet allocation algorithm that learns a topic distribution to represent each document (or text).

[1] http://arnetminer.org/citation (V4 version is used)

[2] http://citeseerx.ist.psu.edu/

Table 1: Average Macro F1 Score and Standard Deviation on **Citeseer-M10** Network

%p	DeepWalk	Doc2Vec	DW+D2V	DNRL	LDA	RTM	LINE	TADW	**TriDNR**
10	0.354±0.007	0.432±0.008	0.495±0.007	0.564±0.007	0.458±0.005	0.504±0.007	0.531±0.004	0.600±0.008	**0.626±0.009**
30	0.411±0.002	0.477±0.005	0.586±0.004	0.667±0.003	0.549±0.003	0.598±0.004	0.569±0.006	0.652±0.005	**0.715±0.004**
50	0.425±0.004	0.494±0.004	0.614±0.006	0.702±0.006	0.581±0.009	0.629±0.005	0.581±0.005	0.671±0.006	**0.753±0.006**
70	0.434±0.007	0.503±0.006	0.628±0.009	0.725±0.005	0.589±0.007	0.638±0.011	0.589±0.010	0.681±0.005	**0.777±0.006**

Table 2: Average Macro F1 Score and Standard Deviation on **DBLP** Network

%p	DeepWalk	Doc2Vec	DW+D2V	DNRL	LDA	RTM	LINE	TADW	**TriDNR**
10	0.398±0.004	0.605±0.005	0.653±0.005	0.663±0.002	0.644±0.002	0.665±0.004	0.427±0.003	0.676±0.006	**0.687±0.004**
30	0.423±0.003	0.617±0.003	0.681±0.003	0.698±0.002	0.652±0.003	0.674±0.001	0.438±0.003	0.689±0.004	**0.727±0.002**
50	0.426±0.002	0.620±0.003	0.686±0.003	0.710±0.002	0.655±0.003	0.676±0.002	0.438±0.002	0.692±0.003	**0.738±0.003**
70	0.428±0.004	0.623±0.003	0.690±0.003	0.718±0.003	0.654±0.003	0.678±0.003	0.439±0.003	0.695±0.003	**0.744±0.002**

5. **RTM** [Chang and Blei, 2009] is the relational topic model which captures both text and network structure to learn topic distributions of each document.
6. **TADW** [Yang *et al.*, 2015] is a state-of-the-art algorithm that utilizes both network and text information to learn the network representation.
7. **DW+D2V** approach simply concatenates the vector representations learned by different neural network models, *i.e.*, DeepWalk and Doc2Vec.
8. **TriDNR** is our proposed tri-party deep network representation algorithm that exploits network structure, node content, label information for learning.
9. **DNRL** is a variant of our TriDNR, which only uses node content and label information (lower layer of Fig 2) and ignores the network structure information.

Measures & Parameter Setting We perform node classification to evaluate the quality of different algorithms. In each network, $p\%$ nodes are randomly labeled, the rest are unlabeled. The whole network, including node content, are used to learn the network representation. Once we obtained the node vectors using different comparing methods, we train a linear SVM from the training data (nodes) to predict unlabeled nodes. We choose a linear classifier, instead of non-linear model or sophisticated relational classifiers [Sen *et al.*, 2008], in order to reduce the impact of complicated learning approaches on the classification performance.

The default parameter for TriDNR are set as follows: window size b=8, dimensions k =300, training size $p = 30\%$, $\alpha = 0.8$. For fairness of comparison, all comparing algorithms will use the same same number of features k. The parameters for other algorithms will keep the same or be close to TriDNR as much as possible. For instance, for all neural network models, we use window size $b = 8$. The rest parameters are set following the suggestion in their original papers. For each parameter setting, we repeat the experiment 10 times and report the average results and standard deviation.

5.2 Performance on Node Classification

We vary the percentages of training samples $p\%$ from 10% to 70% and report the results in Table 1 and Table 2.

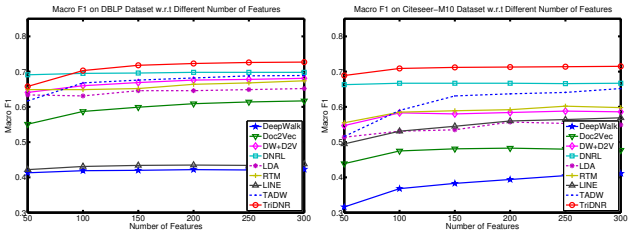


Figure 3: Results with Different Number of Features k.

Table 1 and Table 2 show that DeepWalk and LINE algorithms perform fairly poor on citation networks. This is mainly because the network structure is rather sparse and only contains limited information. Doc2Vec (Paragraph Vectors) is much better than DeepWalk, especially on DBLP dataset, as the text content (title) has rich information comparing to network structure. When concatenating the embeddings from DeepWalk and Doc2Vec, as DW+D2V does, the representation is substantially improved. However, DW+D2V is still far from optimal, comparing with TriDNR algorithm.

Importance of Label Information. Our experimental results show that DNRL, which uses label information to learn the network embedding (only used the lower layer of our TriDNR model), has already significantly outperformed the naive combination of two neural network models (DW+D2V). This observation validates the importance of label information for network representation.

Effectiveness of Coupled design. When we add another layer of neural network on top of DNRL, our proposed TriDNR model shows that it outperforms DeepWalk, Doc2Vec, and DW+D2V by a large margin. The experimental result demonstrated the importance of using label information and the effectiveness of our coupled architecture design.

Topic model based algorithms: As for the topic model based algorithms, relational topic model (RTM) outperforms the traditional topic model LDA. This is because RTM takes the network structure into consideration. Both LDA and RTM are not comparable to TriDNR.

TriDNR *vs.* TADW: The results in Table 1 and 2 demonstrate that TriDNR is also superior to TADW. This is be-

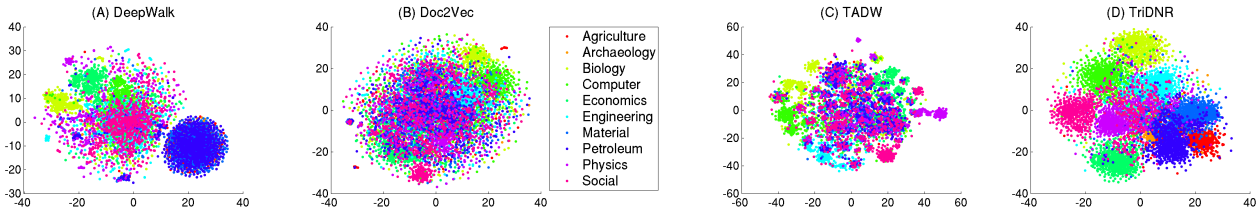


Figure 4: 2D visualization on **Citeseer-M10** network by different algorithms.

cause: (1) TADW is a matrix factorization algorithm which factorizes a matrix M (sum of a series transition matrix) together with a text matrix T. In reality, an accurate M is very computationally expensive so TADW only factorizes an approximate M, which will affect its representation power; (2) TADW does not consider context information, and its T matrix cannot preserve the orders of word; (3) TADW does not explore the valuable label information for a specific task, resulting in suboptimal result only. In contrast, TriDNR takes each document (node) and the words within of a window (including the order information) into consideration, which well preserve the neighborhood relationship and documents. Additionally, TriDNR exploits label information, which significantly improves the representation power, leading to a good classification performance.

Overall, TriDNR significantly outperforms its peers. With $p\%$=70, TriDNR (0.777) beats DeepWalk (0.434) and TADW (0.681) by 79.0% and 14.1% in the Citeseer-M10 network.

Results with different number of dimensions k

We vary the number of dimensions (k) used in comparing algorithms, and report the results in Fig 3. When k increases from 50 to 150, there is a slightly increase for TriDNR algorithm. Afterwards, not much difference is observed with different number of features. The results show that TriDNR is very stable with various dimension size.

5.3 Case Study

In this subsection, we retrieve the most similar nodes *w.r.t.* a given query node. Specifically, we compute the top 5 nearest neighbors with Consine distance based on the vector representations learned by different algorithms. The results are shown in Table 3.

The query paper is the best paper of KDD 2006 and is also one of the top-10 downloaded paper according to ACM digital library [3] (access on 19/01/2016). Basically, the paper considers using advanced optimization techniques like cutting plane algorithm for training linear svm algorithm.

Table 3 shows that all results retrieved by TriDNR are similar or closely related to SVM or optimization techniques. For instance, the first result *optimized cutting plane algorithm for support vector machines* is actually a following-up work of the query paper. In contrast, the results returned by Doc2Vec or TADW only have 1 record matching with the query, which is mostly based on the term *linear time*. For DeepWalk, there are indeed some similar answers by using the network information (citation relationship), because this is a highly cited

[3]http://dl.acm.org/event.cfm?id=RE329

Table 3: Top-5 Similar Node Search: matched results are (•)

Query: Training Linear SVMs in Linear Time [Joachims, 2006]
TriDNR :
1. optimized cutting plane algorithm for support vector machines (•)
2. proximal regularization for online and batch learning (•)
3. a sequential dual method for large scale multi-class linear svms (•)
4. a dual coordinate descent method for large-scale linear svm (•)
5. bootstrap based pattern selection for support vector regression (•)
Doc2Vec :
1. training linear discriminant analysis in linear time (•)
2. circularity measuring in linear time
3. mining association rules in hypertext databases
4. approximate matching in xml
5. disaggregations in databases
DeepWalk :
1. bootstrap based pattern selection for support vector regression (•)
2. large margin training for hidden markov models with ...
3. optimized cutting plane algorithm for support vector machines (•)
4. proximal regularization for online and batch learning (•)
5. extremely fast text feature extraction for classification and indexing
TADW :
1. Circularity Measuring in Linear Time
2. Training Linear Discriminant Analysis in Linear Time (•)
3. Maximal Incrementality in Linear Categorial Deduction
4. Variational Linear Response
5. How Linear are Auditory Cortical Responses

paper (1348 according to Google Scholar by 19/01/2016). However, its result are still worse than our TriDNR algorithm.

5.4 Network Visualization

An important application of network representation is to create meaningful visualizations that layout a network in two dimensional space. Following [Tang *et al.*, 2015], we learn a low-dimensional representation for each node and map the Citeseer-10 network into a 2D space in Fig 4.

In Fig 4, the visualization using Doc2Vec and TADW algorithms are not very meaningful, in which the papers from the same cluster are not clustered together. The result obtained by DeepWalk is slightly better, in which a large group is formed representing the research of *Petroleum Chemistry*. However papers from other groups are still highly overlapping. The visualization results of TriDNR are quite clear, with meaningful layout for each class.

6 Conclusion

In this paper, we proposed a Tri-party deep network representation algorithm. We argued that most exiting algorithms are simple shallow methods that only use one aspect of node information. In addition, none of the existing method is able to utilize the label information in the network for representation. Accordingly, we proposed a coupled neural network based algorithm to exploit inter-node relationships, node-content correlation, and label-content correspondence in a networks to learn an optimal representation for each node in the network. Experimental results demonstrated the superb performance of our algorithms. The key contribution of the paper is twofold: (1) we exploit network representation from multiple parties and different network levels, and (2) we propose a new neural network model for deep network representation learning.

References

[Blei *et al.*, 2003] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[Bordes *et al.*, 2011] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[Chang and Blei, 2009] Jonathan Chang and David M Blei. Relational topic models for document networks. In *International Conference on Artificial Intelligence and Statistics*, pages 81–88, 2009.

[Chang *et al.*, 2015] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. Heterogeneous network embedding via deep architectures. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 119–128. ACM, 2015.

[Joachims, 2006] Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM, 2006.

[Le and Mikolov, 2014] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[Lim and Buntine, 2014] Kar Wai Lim and Wray Buntine. Bibliographic analysis with the citation network topic model. In *Proceedings of the Sixth Asian Conference on Machine Learning*, pages 142–158, 2014.

[Luong *et al.*, 2013] Minh-Thang Luong, Richard Socher, and Christopher D Manning. Better word representations with recursive neural networks for morphology. *CoNLL-2013*, 104, 2013.

[Mikolov *et al.*, 2013] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[Morin and Bengio, 2005] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pages 246–252. Citeseer, 2005.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

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

[Socher *et al.*, 2013] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934, 2013.

[Tang and Liu, 2009] Lei Tang and Huan Liu. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 817–826. ACM, 2009.

[Tang *et al.*, 2008] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 990–998. ACM, 2008.

[Tang *et al.*, 2015] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[Tian *et al.*, 2014] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. Learning deep representations for graph clustering. In *AAAI*, pages 1293–1299, 2014.

[Yang *et al.*, 2015] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. Network representation learning with rich text information. In *International Joint Conference on Artificial Intelligence*, 2015.