# Network Representation Learning with Rich Text Information

**Cheng Yang**[1,2], **Zhiyuan Liu**[1,2*], **Deli Zhao**[2], **Maosong Sun**[1], **Edward Y. Chang**[2]
[1] Department of Computer Science and Technology,
State Key Lab on Intelligent Technology and Systems,
National Lab for Information Science and Technology,
Tsinghua University, Beijing 100084, China
[2] HTC Beijing Advanced Technology and Research Center, China

## Abstract

Representation learning has shown its effectiveness in many tasks such as image classification and text mining. Network representation learning aims at learning distributed vector representation for each vertex in a network, which is also increasingly recognized as an important aspect for network analysis. Most network representation learning methods investigate network structures for learning. In reality, network vertices contain rich information (such as text), which cannot be well applied with algorithmic frameworks of typical representation learning methods. By proving that DeepWalk, a state-of-the-art network representation method, is actually equivalent to matrix factorization (MF), we propose text-associated DeepWalk (TADW). TADW incorporates text features of vertices into network representation learning under the framework of matrix factorization. We evaluate our method and various baseline methods by applying them to the task of multi-class classification of vertices. The experimental results show that, our method outperforms other baselines on all three datasets, especially when networks are noisy and training ratio is small. The source code of this paper can be obtained from https://github.com/albertyang33/TADW.

## 1 Introduction

Networks are ubiquitous in our daily lives, e.g., friendship between Facebook users or citations between academic papers. In recent years, researchers have extensively studied on many important machine learning applications in networks, such as vertex classification [Sen *et al.*, 2008], tag recommendation [Tu *et al.*, 2014], anomaly detection [Bhuyan *et al.*, 2014] and link prediction [Lü and Zhou, 2011]. Data sparsity is the common problem faced by these tasks. To address the sparsity issue, network representation learning (NRL) encodes and represents each vertex in a unified low-dimensional space. NRL facilitates us to better understand the semantic relatedness between vertices, and further alleviates the inconveniences caused by sparsity [Perozzi *et al.*, 2014].

*Corresponding author: Zhiyuan Liu(liuzy@tsinghua.edu.cn)

Most works in NRL learn representations from network structure. For example, social dimensions [Tang and Liu, 2009; 2011] are proposed by computing eigenvectors of Laplacian or modularity matrix of a network. Recently, Skip-Gram, a word representation model in NLP, is introduced to learn vertex representations from random walk sequences in social networks, dubbed DeepWalk [Perozzi *et al.*, 2014]. Both social dimensions and DeepWalk methods take a network structure as input to learn network representations, without considering any other information.

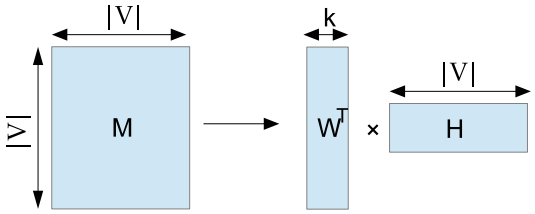
In real world, a vertex in a network usually has rich information, such as text content and other meta data. For example, Wikipedia articles connect to each other and form a network, and each article, as a vertex, has substantial text information, which may also be important to NRL. Hence, we come up with an idea to learn network representations from both network structure and text information.

A straightforward method is to learn representations from text features and network features independently, and then concatenate the two separate representations. The method, however, does not take the sophisticated interactions between network structure and text information into consideration, and thus usually leads to no avail. It is also non-trivial to incorporate text information in existing NRL frameworks. For example, DeepWalk cannot easily handle additional information during its random walks in a network.

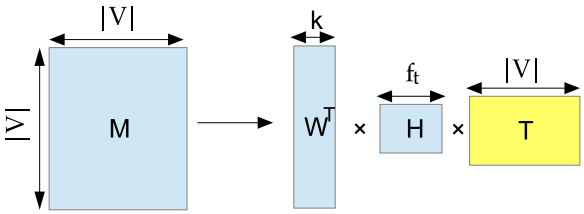
Fortunately, given a network $G = (V, E)$, we prove that DeepWalk is actually factorizing a matrix $M \in \mathbb{R}^{|V|\times|V|}$ where each entry $M_{ij}$ is logarithm of the average probability that vertex $v_i$ randomly walks to vertex $v_j$ in fixed steps. Figure 1(a) shows the MF-style DeepWalk: factorize matrix $M$ into the product of two low-dimensional matrices $W \in \mathbb{R}^{k\times|V|}$ and $H \in \mathbb{R}^{k\times|V|}$ where $k \ll |V|$. DeepWalk then takes the matrix $W$ as vertex representation. We will give a detailed introduction in the next section.

The matrix-factorization view of DeepWalk inspires us to introduce text information into MF for NRL. Figure 1(b) shows the main idea of our method: factorize matrix $M$ into the product of three matrices: $W \in \mathbb{R}^{k\times|V|}$, $H \in \mathbb{R}^{k\times f_t}$ and text features $T \in \mathbb{R}^{f_t\times|V|}$. Then we concatenate $W$ and $HT$ as $2k$-dimensional representations of vertices.

We test our algorithm against several baselines on three datasets. The classification accuracy of our representation outperforms other baselines by at most $2\%$ to $10\%$ when




(a) DeepWalk




(b) TADW

Figure 1: (a) DeepWalk as matrix factorization. (b) Text-associated matrix factorization (TADW).

the ratio of training set ranges from $10\%$ to $50\%$. We also test these methods with a semi-supervised classifier, Transductive SVM (TSVM), when training ratio is less than $10\%$. Our method has a $5\%$ to $20\%$ advantage than other baselines with $1\%$ training ratio, especially when network information is noisy.

There are two main contributions of this paper: (1) We prove that DeepWalk algorithm actually factorizes a matrix $M$ and figure out the closed form of $M$. (2) We introduce text features into NRL and get a $5\%$ to $20\%$ advantage as compared to other baselines especially when the training ratio is small.

**Related Work** Representation learning is widely used in computer vision [Krizhevsky *et al.*, 2012], natural language processing [Mikolov *et al.*, 2013] and knowledge representation learning [Lin *et al.*, 2015]. Some researches focus on NRL [Chen *et al.*, 2007; Tang and Liu, 2009; 2011; Perozzi *et al.*, 2014], but none of them can be generalized to deal with other features of vertices trivially. To the best of our knowledge, little work has been devoted to consider text information in NRL. There are some topic models, such as NetPLSA [Mei *et al.*, 2008], considering both networks and text information for topic modeling, in which we can represent each vertex with a topic distribution. In this paper, we take NetPLSA as a baseline method.

The rest of this paper is organized as follows. Section 2 gives the formal definition for NRL and proves that DeepWalk is actually equivalent to matrix factorization. Section 3 presents our algorithm for NRL with text features. We introduce the datasets and experimental results in Section 4. Section 5 concludes the paper.

## 2 DeepWalk as Matrix Factorization

### 2.1 Formalization of NRL

Network representation learning is formalized as follows. Given a network $G = (V, E)$, we want to build a low-dimensional representation $r_v \in \mathbb{R}^k$ for each vertex $v$, where $k$ is expected to be much smaller than $|V|$.

As a dense real-valued representation, $r_v$ can alleviate the sparsity of network representations such as adjacency matrix. We can regard $r_v$ as features of vertex $v$ and apply the features to many machine learning tasks like vertex classification. The features can be conveniently fed to many classifiers, e.g. logistic regression and SVM. Also note that the representation is not task-specific and can be shared among different tasks.

We first introduce DeepWalk and then give the proof of equivalence between DeepWalk and matrix factorization.

### 2.2 DeepWalk

DeepWalk introduced Skip-Gram [Mikolov *et al.*, 2013], a widely-used distributed word representation method, into the study of social network for the first time to learn vertex representation according to network structure.

DeepWalk first generates short random walks which have been used as a similarity measure [Fouss *et al.*, 2007]. Given a sequence of vertices $S = \{v_1, v_2, \ldots, v_{|S|}\}$ generated by random walks, we regard the vertices $v \in \{v_{i-t}, \ldots, v_{i+t}\} \setminus \{v_i\}$ as the context of the center vertex $v_i$, where $t$ is the window size. Following the idea of Skip-Gram, DeepWalk aims to maximize the average log probability of all vertex-context pairs in the random walk vertex sequence $S$:

$$\frac{1}{|S|}\sum_{i=1}^{|S|}\sum_{-t\le j\le t, j\neq 0}\log p(v_{i+j}|v_i), \tag{1}$$

where $p(v_j|v_i)$ is defined by softmax function,

$$p(v_j|v_i) = \frac{\exp(c_{v_j}^T r_{v_i})}{\sum_{v\in V}\exp(c_v^T r_{v_i})}. \tag{2}$$

Here $r_{v_i}$ and $c_{v_j}$ are the representation vectors of the center vertex $v_i$ and its context vertex $v_j$. Namely, each vertex $v$ has two representation vectors: $r_v$ when $v$ plays as a center vertex and $c_v$ when $v$ plays as a context vertex.

Afterwards, DeepWalk uses Skip-Gram and Hierarchical Softmax to learn representations of vertices from sequences generated by random walks. Note that Hierarchical Softmax [Morin and Bengio, 2005] is a variant of softmax for speedup.

### 2.3 Equivalence Proof

Suppose a vertex-context set $D$ is generated from random walk sequences, where each member of $D$ is a vertex-context pair $(v, c)$. $V$ is the set of vertices and $V_C$ is the set of context vertices. In most cases, $V = V_C$.

DeepWalk embeds a vertex $v$ into a $k$-dimensional vector $r_v \in \mathbb{R}^k$. Also, a context vertex $v \in V_C$ is represented by a $k$-dimensional vector $c_v \in \mathbb{R}^k$. Let $W$ be a $k \times |V|$ matrix where column $i$ is vector $r_{v_i}$ and $H$ be a $k \times |V_C|$ matrix where column $j$ is vector $c_{v_j}$. Our goal is to figure out the closed form of matrix $M$ where $M = W^T H$.

Let us consider a vertex-context pair $(v, c)$. $N(v, c)$ denotes the number of times that $(v, c)$ appears in $D$. $N(v) = \sum_{c' \in V_C} N(v, c')$ and $N(c) = \sum_{v' \in V} N(v', c)$ denote the numbers of times $v$ and $c$ appear in $D$, respectively.

It has been shown that Skip-Gram with Negative Sampling (SGNS) is implicitly factorizing a word-context matrix $M$ [Levy and Goldberg, 2014] by assuming that dimensionality $k$ is sufficiently large. Each entry in $M$ is

$$M_{ij} = \log \frac{N(v_i, c_j) \cdot |D|}{N(v_i) \cdot N(c_j)} - \log n, \tag{3}$$

where $n$ is the number of negative samples for each word-context pair. $M_{ij}$ can be interpreted as Pointwise Mutual Information (PMI) of word-context pair $(v_i, c_j)$ shifted by $\log n$. Similarly, we can prove that Skip-Gram with softmax is factorizing a matrix $M$ where

$$M_{ij} = \log \frac{N(v_i, c_j)}{N(v_i)}. \tag{4}$$

We now discuss what $M_{ij}$ in DeepWalk represents. It is clear that the method of sampling vertex-context pairs will affect matrix $M$. Assume that the network is connected and undirected and that window size is $t$. We will discuss $N(v)/|D|$, $N(c)/|D|$ and $N(v, c)/N(v)$ based on an ideal sampling method for DeepWalk algorithm: first we generate a sufficiently long random walk $RW$. Denote $RW_i$ as the vertex on position $i$ of $RW$. Then we add vertex-context pair $(RW_i, RW_j)$ into $D$ if and only if $0 < |i - j| \leq t$.

Each occurrence of vertex $i$ will be recorded $2t$ times in $D$ for undirected graph. Thus $N(v_i)/|D|$ is the frequency of $v_i$ appears in the random walk, which is exactly the PageRank value of $v_i$. Also note that $2tN(v_i, v_j)/N(v_i)$ is the expectation times that $v_j$ is observed in left or right $t$ neighbors of $v_i$. Now we try to figure out $N(v_i, v_j)/N(v_i)$ based on this comprehension.

Denote the transition matrix in PageRank as $A$. Let $d_i$ be the degree of vertex $i$, and we have $A_{ij} = 1/d_i$ if $(i, j) \in E$ and $A_{ij} = 0$ otherwise. We use $e_i$ to denote a $|V|$-dimensional row vector where all entries are 0 except the $i$-th entry is 1.

Suppose that we start a random walk from vertex $i$ and use $e_i$ to denote the initial state. Then $e_i A$ is the distribution over all the vertices and the $j$-th entry of $e_i A$ is the probability that vertex $i$ walks to vertex $j$. Hence the $j$-th entry of $e_i A^t$ is the probability that vertex $i$ walks to vertex $j$ at exactly $t$ steps where $A^t$ is the multiplication of matrix $A$ by $t$ times. Thus $[e_i(A + A^2 + \cdots + A^t)]_j$ is the expectation times that $v_j$ appears in right $t$ neighbors of $v_i$. Hence we have

$$\frac{N(v_i, v_j)}{N(v_i)} = \frac{[e_i(A + A^2 + \cdots + A^t)]_j}{t}. \tag{5}$$

This equality also holds for directed graph. Hence, we can see $M_{ij} = \log N(v_i, v_j)/N(v_i)$ is logarithm of the average probability that vertex $i$ randomly walks to vertex $j$ in $t$ steps.

By proving that DeepWalk is equivalent to matrix factorization, we propose to incorporate rich text information for NRL based on DeepWalk-derived matrix factorization.

# 3 Our Method

In this section, we first give a brief introduction to low-rank matrix factorization, and then we formulate our method of representation learning from both network and text information.

## 3.1 Low-rank Matrix Factorization

Matrix is a common way to represent relational data. An interesting topic for matrix analysis is to figure out the inherent structure of a matrix by a fraction of its entries. One assumption is that matrix $M \in \mathbb{R}^{b \times d}$ admits an approximation of low rank $k$, where $k \ll \{b, d\}$. Then we can complete the missing entries in matrix $M$ with such a low-rank approximation under this assumption. However, solving a rank constraint optimization is always NP-hard. Therefore researchers resort to finding matrices $W \in \mathbb{R}^{k \times b}$ and $H \in \mathbb{R}^{k \times d}$ to minimize the loss function $L(M, W^T H)$ with a trace norm constraint, which is further removed by adding a penalty term to the loss function [Yu *et al.*, 2014]. In this paper, we use square loss function.

Formally, let the observation set of matrix $M$ be $\Omega$. We want to find matrices $W \in \mathbb{R}^{k \times b}$ and $H \in \mathbb{R}^{k \times d}$ to minimize

$$\min_{W,H} \sum_{(i,j) \in \Omega} \big(M_{ij} - (W^T H)_{ij}\big)^2 + \frac{\lambda}{2}\big(||W||_F^2 + ||H||_F^2\big), \tag{6}$$

where $||\cdot||_F$ means Frobenius norm of the matrix and $\lambda$ is a harmonic factor to balance two components.

Low-rank matrix factorization completes matrix $M$ only based on the low-rank assumption of $M$. If items in matrix $M$ have additional features, we can apply inductive matrix completion [Natarajan and Dhillon, 2014] to take advantage of them. Inductive matrix completion utilizes more information of row and column units by incorporating two feature matrices into the objective function. Suppose that we have feature matrices $X \in \mathbb{R}^{f_x \times b}$ and $Y \in \mathbb{R}^{f_y \times d}$ where column $i$ of $X$ and $Y$ are $f_x$ and $f_y$ dimensional feature vectors of unit $i$, respectively. Our goal is to solve matrices $W \in \mathbb{R}^{k \times f_x}$ and $H \in \mathbb{R}^{k \times f_y}$ to minimize

$$\min_{W,H} \sum_{(i,j) \in \Omega} \big(M_{ij} - (X^T W^T H Y)_{ij}\big)^2 + \frac{\lambda}{2}\big(||W||_F^2 + ||H||_F^2\big). \tag{7}$$

Note that, inductive matrix completion is originally proposed to complete gene-disease matrix with gene and disease features [Natarajan and Dhillon, 2014], the goal of which is quite different from that of our work. Inspired by the idea of inductive matrix completion, we introduce text information into NRL.

## 3.2 Text-Associated DeepWalk (TADW)

Given a network $G = (V, E)$ and its corresponding text feature matrix $T \in \mathbb{R}^{f_t \times |V|}$, we propose text-associated DeepWalk (TADW) to learn representation of each vertex $v \in V$ from both network structure $G$ and text features $T$.

Recall that DeepWalk factorizes matrix $M$ where $M_{ij} = \log([e_i(A + A^2 + \cdots + A^t)]_j/t)$. Computing an accurate $M$ has the complexity of $O(|V|^3)$ when $t$ gets large. In fact, DeepWalk uses a sampling method based on random walk to

avoid explicitly computing accurate matrix $M$. When DeepWalk samples more walks, the performance will be better while DeepWalk will be less efficient.

In TADW, we find a tradeoff between speed and accuracy: factorize the matrix $M = (A + A^2)/2$. Here, we factorize $M$ instead of $\log M$ for computational efficiency. The reason is that, $\log M$ has much more non-zero entries than $M$, and the complexity of matrix factorization with square loss is proportional to the number of non-zero elements in matrix $M$ [Yu *et al.*, 2014]. Since most real-world networks are sparse, i.e. $O(E) = O(V)$, computing matrix $M$ takes $O(|V|^2)$ time. If a network is dense, we can even directly factorize matrix $A$. Our task is to solve matrices $W \in \mathbb{R}^{k \times |V|}$ and $H \in \mathbb{R}^{k \times f_t}$ to minimize

$$\min_{W,H} ||M - W^T HT||_F^2 + \frac{\lambda}{2}(||W||_F^2 + ||H||_F^2). \quad (8)$$

For optimizing $W$ and $H$, we alternately minimize $W$ and $H$ because it is a convex function for either $W$ or $H$. Though TADW may converge to a local minimum instead of the global minimum, our method works well in practice as shown in our experiments.

Different from low-rank matrix factorization and inductive matrix completion which focus on completing the matrix $M$, the goal of TADW is to incorporate text features to obtain better network representations. Also, inductive matrix completion obtains matrix $M$ directly from raw data while we artificially build matrix $M$ from the derivation of MF-style DeepWalk. Since both $W$ and $HT$ obtained from TADW can be regarded as low-dimensional representations of vertices, we build a unified $2k$-dimensional matrix for network representations by concatenating them. In experiments, we will show that the unified representation significantly outperforms a naive combination of network representations and text features (i.e. the matrix $T$).

### 3.3 Complexity Analysis

In TADW, the procedure of computing $M$ takes $O(|V|^2)$ time. We use the fast procedure introduced by [Yu *et al.*, 2014] to solve the optimization problem in Equation (8). The complexity of each iteration of minimizing $W$ and $H$ is $O(\text{nnz}(M)k + |V|f_t k + |V|k^2)$ where $\text{nnz}(\cdot)$ indicates the number of non-zero entries. For comparison, the complexity of traditional matrix factorization, i.e. optimization problem in Equation (6), is $O(\text{nnz}(M)k + |V|k^2)$. In our experiments, the optimization converges in 10 iterations.

## 4 Experiments

We use multi-class vertex classification to evaluate the quality of NRL. Formally, we regard low-dimensional representation $\mathcal{R} = \{r_1, r_2, \ldots, r_{|V|}\}$ as vertex features. Our task is to predict the labels of unlabeled set $U$ with labeled set $L$ based on vertex features $\mathcal{R}$.

A number of classifiers in machine learning can deal with this task. We select SVM and transductive SVM for supervised and semi-supervised learning and testing, respectively. Note that, since the representation learning procedure ignores vertex labels in training set, representation learning is unsupervised.

We evaluate TADW with five baseline methods of representation learning using three publicly available datasets [1]. We learn representations from the links or citations between the documents as well as the term frequency-inverse document frequency (TFIDF) matrices of these documents.

### 4.1 Datasets and Experiment Settings

**Datasets**

**Cora** contains $2,708$ machine learning papers from seven classes and $5,429$ links between them. The links are citation relationships between the documents. Each document is described by a binary vector of $1,433$ dimensions indicating the presence of the corresponding word.

**Citeseer** contains $3,312$ publications from six classes and $4,732$ links between them. Similar to Cora, the links are citation relationships between the documents and each paper is described by a binary vector of $3,703$ dimensions.

**Wiki** contains $2,405$ documents from $19$ classes and $17,981$ links between them. The TFIDF matrix of this dataset has $4,973$ columns.

The documents in Cora and Citeseer are short texts generated from titles and abstracts. Stop words and all words with document frequency less than 10 are removed. Each document has 18 or 32 words on average correspondingly. The documents in Wiki are long texts. We remove all documents which have no connection in the network. Each document has $640$ words on average. We regard the networks as undirected graphs.

**TADW Settings**

For all three datasets, we reduce the dimension of word vectors to 200 via SVD decomposition of the TFIDF matrix, and obtain text feature matrices $T \in \mathbb{R}^{200 \times |V|}$. The preprocessing will reduce the number of parameters in matrix $H$. We also take text feature matrix $T$ as a content-only baseline. We select $k = 80$ and $\lambda = 0.2$ for Cora and Citeseer datasets; $k = 100, 200$ and $\lambda = 0.2$ for Wiki dataset. Note that, the dimension of representation vectors from TADW is $2k$.

**Baseline Methods**

**DeepWalk.** DeepWalk [Perozzi *et al.*, 2014] is a network-only representation learning method. We set parameters as follows, walks per vertex $\gamma = 80$ and window size $t = 10$ which are the same with those in the original paper. We choose representation dimension $k = 100$ for Cora and Citeseer and $k = 200$ for Wiki which are the lengths with best performance between 50 and 200.

We also evaluate the performance of MF-style DeepWalk by solving Equation (6) and concatenate $W$ and $H$ as vertex representations. The result is competitive with DeepWalk. Hence we only report the performance of original DeepWalk.

**PLSA.** We use PLSA [Hofmann, 1999] to train a topic model from the TFIDF matrix by regarding each vertex as a document. Hence, PLSA is a content-only baseline. PLSA estimates topic distribution of documents and word distribution of topics via EM algorithm. We use topic distribution of documents as vertex representations.

[1] http://linqs.cs.umd.edu/projects//projects/lbc/index.html.

Table 1: Evaluation results on Cora dataset.

| Classifier | Transductive SVM | | | | SVM | | | | |
|---|---|---|---|---|---|---|---|---|---|
| % Labeled Nodes | 1% | 3% | 7% | 10% | 10% | 20% | 30% | 40% | 50% |
| DeepWalk | 62.9 | 68.3 | 72.2 | 72.8 | 76.4 | 78.0 | 79.5 | 80.5 | 81.0 |
| PLSA | 47.7 | 51.9 | 55.2 | 60.7 | 57.0 | 63.1 | 65.1 | 66.6 | 67.6 |
| Text Features | 33.0 | 43.0 | 57.1 | 62.8 | 58.3 | 67.4 | 71.1 | 73.3 | 74.0 |
| Naive Combination | 67.4 | 70.6 | 75.1 | 77.4 | 76.5 | 80.4 | 82.3 | 83.3 | 84.1 |
| NetPLSA | 65.7 | 67.9 | 74.5 | 77.3 | 80.2 | 83.0 | 84.0 | 84.9 | 85.4 |
| TADW | **72.1** | **77.0** | **79.1** | **81.3** | **82.4** | **85.0** | **85.6** | **86.0** | **86.7** |

Table 2: Evaluation results on Citeseer dataset.

| Classifier | Transductive SVM | | | | SVM | | | | |
|---|---|---|---|---|---|---|---|---|---|
| % Labeled Nodes | 1% | 3% | 7% | 10% | 10% | 20% | 30% | 40% | 50% |
| DeepWalk | - | - | 49.0 | 52.1 | 52.4 | 54.7 | 56.0 | 56.5 | 57.3 |
| PLSA | 45.2 | 49.2 | 53.1 | 54.6 | 54.1 | 58.3 | 60.9 | 62.1 | 62.6 |
| Text Features | 36.1 | 49.8 | 57.7 | 62.1 | 58.3 | 66.4 | 69.2 | 71.2 | 72.2 |
| Naive Combination | 39.0 | 45.7 | 58.9 | 61.0 | 61.0 | 66.7 | 69.1 | 70.8 | 72.0 |
| NetPLSA | 45.4 | 49.8 | 52.9 | 54.9 | 58.7 | 61.6 | 63.3 | 64.0 | 64.7 |
| TADW | **63.6** | **68.4** | **69.1** | **71.1** | **70.6** | **71.9** | **73.3** | **73.7** | **74.2** |

**Text Features.** We use text feature matrix $T \in \mathbb{R}^{200\times|V|}$ as a 200-dimensional representation. The method of Text Features is a content-only baseline.

**Naive Combination.** We can simply concatenate the vectors from both Text Features and DeepWalk for network representations. It has a length of 300 for Cora and Citeseer and 400 for Wiki.

**NetPLSA.** [Mei *et al.*, 2008] proposed to learn topic distributions of documents by considering links between documents as a network regularization that linked documents should share similar topic distributions. We use the network-enhanced topic distribution of documents as network representations. NetPLSA can be regarded as an NRL method considering both network and text information. We set topic numbers to 160 for Cora and Citeseer, and 200 for Wiki.

**Classifiers and Experiment Setup**

For supervised classifier, we use linear SVM implemented by Liblinear [Fan *et al.*, 2008]. For semi-supervised classifiers, we use transductive SVM implemented by SVM-Light [Joachims, 1999]. We use linear kernel for TSVM. We train a one-vs-rest classifier for each class and select the classes with maximum scores in linear SVM and transductive SVM.

We take representations of vertices as features to train classifiers, and evaluate classification accuracy with different training ratios. The training ratio varies from 10% to 50% for linear SVM and 1% to 10% for TSVM. For each training ratio, we randomly select documents as training set and the remaining documents as test set. We repeat the trial for 10 times and report the average accuracy.

## 4.2 Experimental Results and Analysis

Table 1, Table 2 and Table 3 show classification accuracies on Cora, Citeseer and Wiki datasets. Here "-" indicates TSVM can not converge in 12 hours because of low quality of representation (TSVM can always converge in 5 minutes for TADW). We did not show the results of semi-supervised learning on Wiki dataset because supervised SVM has already attained a competitive and even better performance with small training ratio on this dataset. Thus we only report the results of supervised SVM for Wiki. Wiki has much more classes than the other two datasets, which requires more data for sufficient training, hence we set the minimum training ratio to 3%. From these tables, we have following observations:

(1) TADW consistently outperforms all the other baselines on all three datasets. Furthermore, TADW can beat other baselines with 50% less training data on Cora and Citeseer datasets. These experiments demonstrate that TADW is effective and robust.

(2) TADW has more significant improvement for semi-supervised learning. TADW outperforms the best baseline, i.e. naive combination, by 4% on Cora and 10% to 20% on Citeseer. This is because the quality of network representations is poor on Citeseer, while TADW is more robust for learning from noisy data than naive combination.

(3) TADW has an encouraging performance when training ratio is small. The accuracies of most baselines drop quickly as training ratio decreases because their vertex representations are much noisy and inconsistent for training and testing. Instead, since TADW learns representation jointly from both network and text information, the representations have less noises and are more consistent.

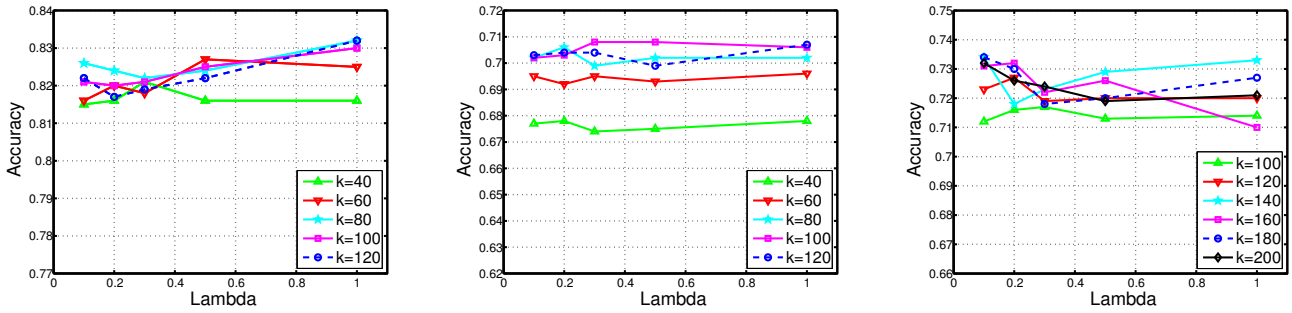
These observations demonstrate the high quality of representations generated by TADW. Moreover, TADW is not task-specific and the representations can be conveniently used for different tasks, such as link prediction, similarity computation and vertex classification. The classification accuracy of TADW is also competitive with several recent collective classification algorithms [Shi *et al.*, 2011; McDowell and Aha, 2012; 2013] though we don't perform specific optimization for the tasks when we learn representations.

## 4.3 Parameter Sensitivity

TADW has two hyperparameters: dimension $k$ and weight of regularization term $\lambda$. We fix training ratio to 10% and test classification accuracies with different $k$ and $\lambda$.

Table 3: Evaluation results on Wiki dataset.

| Classifier | SVM | | | | | | |
|---|---|---|---|---|---|---|---|
| % Labeled Nodes | 3% | 7% | 10% | 20% | 30% | 40% | 50% |
| DeepWalk | 48.4 | 56.6 | 59.3 | 64.3 | 66.2 | 68.1 | 68.8 |
| PLSA | 58.3 | 66.5 | 69.0 | 72.5 | 74.7 | 75.5 | 76.0 |
| Text Features | 46.7 | 60.8 | 65.1 | 72.9 | 75.6 | 77.1 | 77.4 |
| Naive Combination | 48.7 | 62.6 | 66.3 | 73.0 | 75.2 | 77.1 | 78.6 |
| NetPLSA | 56.3 | 64.6 | 67.2 | 70.6 | 71.7 | 71.9 | 72.3 |
| TADW (k=100) | **59.8** | **68.2** | **71.6** | **75.4** | **77.3** | **77.7** | **79.2** |
| TADW (k=200) | **60.4** | **69.9** | **72.6** | **77.3** | **79.2** | **79.9** | **80.3** |




(a) Cora Dataset (b) Citeseer Dataset (c) Wiki Dataset

Figure 2: Parameter sensitivity of $k$ and $\lambda$

We let $k$ vary from 40 to 120 and $\lambda$ vary from 0.1 to 1 for Cora and Citeseer datasets; $k$ vary from 100 to 200 and $\lambda$ vary from 0.1 to 1 for Wiki dataset. Figure 2 shows the variation of classification accuracies with different $k$ and $\lambda$. The accuracies range within 1.5%, 1% and 2% for fixed $k$ on Cora, Citeseer and Wiki, respectively. The accuracies are competitive when $k \geq 80$ on Cora and Citeseer and $k \geq 140$ on Wiki. Therefore TADW can keep stable when $k$ and $\lambda$ vary within a reasonable range.

### 4.4 Case Study

To better understand the effectiveness of text information for NRL, we present an example in Cora dataset. The document title is "Irrelevant Features and the Subset Selection Problem". We call this paper IFSSP for short. The class label of IFSSP is "Theory". As shown in Table 4, using representations generated by DeepWalk and TADW, we find 5 most similar documents of IFSSP ranked by cosine similarity.

We find that, all these documents are cited by IFSSP. However, 3 of the 5 documents found by DeepWalk have different class labels while the first 4 documents found by TADW have the same label "Theory". This indicates that, as compared to pure network-based DeepWalk, TADW can learn better network representations with the help of text information.

The 5th document found by DeepWalk also shows another limitation of considering only network information. "MLC Tutorial A Machine Learning library of C classes" (MLC for short) is a document describing a general toolbox, which may be cited by many works in different topics. Once some of these works cite IFSSP as well, DeepWalk will tend to give IFSSP a similar representation with MLC even though they are totally on different topics.

Table 4: Five nearest documents by DeepWalk and TADW

| Top 5 nearest documents by DeepWalk | |
|---|---|
| Title | Class Label |
| Feature selection methods for classifications | Neural Network |
| Automated model selection | Rule Learning |
| Compression-Based Feature Subset Selection | Theory |
| Induction of Condensed Determinations | Case Based |
| MLC Tutorial A Machine Learning library of C classes | Theory |
| **Top 5 nearest documents by TADW** | |
| Title | Class Label |
| Feature subset selection as search with probabilistic estimates | Theory |
| Compression-Based Feature Subset Selection | Theory |
| Selection of Relevant Features in Machine Learning | Theory |
| NP-Completeness of Searches for Smallest Possible Feature Sets | Theory |
| Feature subset selection using a genetic algorithm | Genetic Algorithms |

## 5 Conclusion

In this paper we prove that DeepWalk, a typical NRL algorithm, is actually equivalent to matrix factorization. Based on the view of matrix factorization, we propose a novel NRL

method TADW to incorporate text features of vertices into network representation learning. Experimental results on three datasets with different training ratios show the effectiveness and robustness of TADW as compared to other baselines.

TADW can also be regarded as a framework for combining features obtained from two different types of information. Rather than simply concatenating features, TADW provides a novel view of feature combination by jointly modeling them via matrix factorization.

For future work, an intriguing direction is to explore the online and distributed learning of TADW for the scenario of large-scale network data. Also, we can investigate other technics of matrix factorization such as matrix co-factorization to involve rich information from other sources.

## Acknowledgments

This work is supported by the 973 Program (No. 2014CB340501), the National Natural Science Foundation of China (NSFC No. 61202140) and HTC University Research Awards Program.

## References

[Bhuyan *et al.*, 2014] Monowar H Bhuyan, DK Bhattacharyya, and Jugal K Kalita. Network anomaly detection: methods, systems and tools. *IEEE Communications Surveys & Tutorials*, 16(1):303–336, 2014.

[Chen *et al.*, 2007] Mo Chen, Qiong Yang, and Xiaoou Tang. Directed graph embedding. In *Proceedings of IJCAI*, pages 2707–2712, 2007.

[Fan *et al.*, 2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.

[Fouss *et al.*, 2007] Francois Fouss, Alain Pirotte, J-M Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE TKDE*, 19(3):355–369, 2007.

[Hofmann, 1999] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of SIGIR*, pages 50–57, 1999.

[Joachims, 1999] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 169–184. MIT Press, Cambridge, MA, 1999.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of NIPS*, pages 1097–1105, 2012.

[Levy and Goldberg, 2014] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Proceedings of NIPS*, pages 2177–2185, 2014.

[Lin *et al.*, 2015] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI*, 2015.

[Lü and Zhou, 2011] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

[McDowell and Aha, 2012] Luke McDowell and David Aha. Semi-supervised collective classification via hybrid label regularization. In *Proceedings of ICML*, 2012.

[McDowell and Aha, 2013] Luke K McDowell and David W Aha. Labels or attributes?: rethinking the neighbors for collective classification in sparsely-labeled networks. In *Proceedings of CIKM*, pages 847–852, 2013.

[Mei *et al.*, 2008] Qiaozhu Mei, Deng Cai, Duo Zhang, and ChengXiang Zhai. Topic modeling with network regularization. In *Proceedings of WWW*, pages 101–110, 2008.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119, 2013.

[Morin and Bengio, 2005] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of AISTATS*, volume 5, pages 246–252, 2005.

[Natarajan and Dhillon, 2014] Nagarajan Natarajan and Inderjit S Dhillon. Inductive matrix completion for predicting gene–disease associations. *Bioinformatics*, 30(12):i60–i68, 2014.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of SIGKDD*, pages 701–710, 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.

[Shi *et al.*, 2011] Xiaoxiao Shi, Yao Li, and Philip Yu. Collective prediction with latent graphs. In *Proceedings of CIKM*, pages 1127–1136, 2011.

[Tang and Liu, 2009] Lei Tang and Huan Liu. Relational learning via latent social dimensions. In *Proceedings of SIGKDD*, pages 817–826, 2009.

[Tang and Liu, 2011] Lei Tang and Huan Liu. Leveraging social media networks for classification. *Proceedings of SIGKDD*, 23(3):447–478, 2011.

[Tu *et al.*, 2014] Cunchao Tu, Zhiyuan Liu, and Maosong Sun. Inferring correspondences from multiple sources for microblog user tags. In *Proceedings of SMP*, pages 1–12, 2014.

[Yu *et al.*, 2014] Hsiang-Fu Yu, Prateek Jain, and Inderjit S Dhillon. Large-scale multi-label learning with missing labels. In *Proceedings of ICML*, 2014.