Knowledge-Based Systems 220 (2021) 106917

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

ETINE: Enhanced Textual Information Network Embedding

Wenfeng Liu [a], Maoguo Gong [a,*], Zedong Tang [b]

[a] School of Electronic Engineering, Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, Xidian University, Xi'an, Shaanxi Province 710071, China
[b] Academy of Advanced Interdisciplinary Research, Xidian University, Xi'an, Shaanxi Province 710071, China

ARTICLE INFO

Article history:
Received 14 August 2020
Received in revised form 26 January 2021
Accepted 1 March 2021
Available online 6 March 2021

Keywords:
Network embedding
Textual information networks
Structural proximity
Textual proximity
Matrix approximation

ABSTRACT

In many reality networks, nodes contain rich text attribute information that exhibits significant role in describing the properties of them and relationship between them. The integration of structural and textual information is beneficial to downstream network analysis tasks. In this work, we present an Enhanced Textual Information Network Embedding model, called ETINE, for learning network embeddings with not only global structural information but also deep semantic relationship between nodes. Specifically, we formulate the optimization of our proposed structure-based and text-based loss functions as a matrix approximation problem. Moreover, to enhance the efficiency and robustness of the proposed method, we propose to optimize the loss functions with an efficient randomized singular value decomposition (RSVD) method. Extensive experiments on four benchmarks demonstrate that our model outperforms other state-of-the-art baselines in multi-class node classification, network reconstruction and node clustering tasks.

1. Introduction

Networks are ubiquitous in numerous real-world scenarios, such as social networks in social media sites [1,2], citation networks in academia [3], protein–protein interaction networks in biology [4], etc. Most network analytic methods (e.g., logistic regression, SVM) heavily rely on the representations of networks. Network embedding, also called network representation learning, has gradually becoming the hottest research topic in recent years [5–10]. Network embedding aims at storing and learning the topology, node attributes and/or other side information of a network into a low-dimensional embedding space. The learned network embeddings can be easily and efficiently applied to the downstream machine learning applications, such as network reconstruction [11,12], node clustering [13] and multi-class node classification [14–16].

Currently, most existing network embedding approaches mainly focus on explicit global and/or local structure information of a network [17–21]. In reality, network vertices contain rich attribute information (e.g., text, label, etc.), which play an important role in exhibiting the interdependency among vertices. Although topological structure and node data explicitly and implicitly define distinct relational information, they complement each other essentially and jointly describe the information networks. Firstly, when the network topology is closely related to the network data (i.e., they appear consistency relations), according to the study of network homophily, two nodes with stronger correlation of network topology tend to have similar node data. Take the network in social media sites (e.g., Facebook, Twitter) as an example, the contents published by two users who pay close attention to each other are often similar. Another example is the network in the scientific literature, the papers with close citing relationship often belong to relevant research fields. Secondly, current complex networks are always large-scale and highly sparse, network embedding methods that only consider sparse network topology are incapable of representing the network property comprehensively. Node data, thus can be served as rich and important auxiliary information for network embedding. Fig. 1 shows a toy example of Facebook's social network, in which users build directed connections by following others and share their own research topics. As can be seen from the figure, the research fields of these four users are natural language processing (NLP), and there are many research topics including language modeling, speech recognition, question answering system, etc. Although there exist following–followee relationships among the four users, their research topics are not the same. For example, *user A* follows *user C*, but they have no common research topics. Additionally, there is no following–followee relationship between *user B* and *user C*, but they share common topic about language modeling. Thus, apart from the topology information, node attribute (i.e., text) also contains rich and important information which is critical for learning high-quality and comprehensive node representations.

* Corresponding author.
E-mail address: gong@ieee.org (M. Gong).

https://doi.org/10.1016/j.knosys.2021.106917
0950-7051/

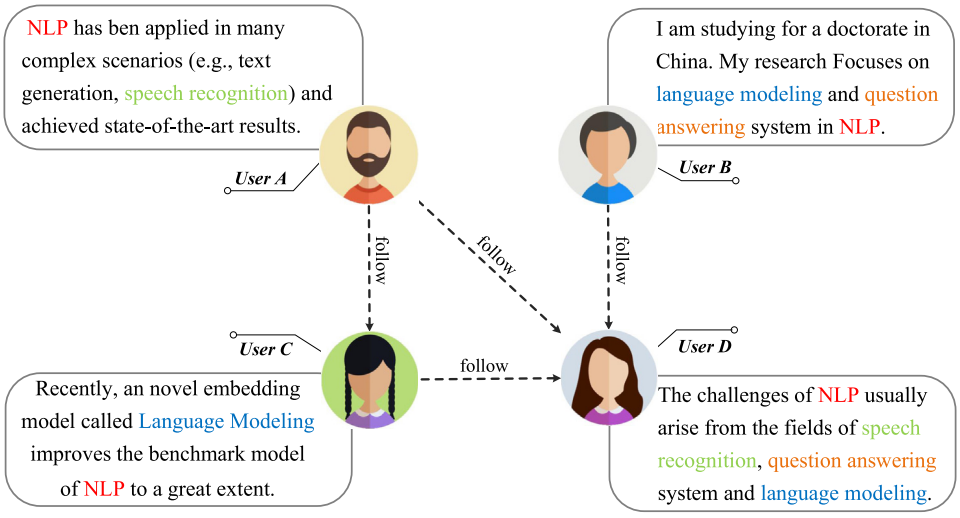


Fig. 1. This is an illustration of textural information networks (TINs) via Facebook's social network.(colored fonts represent common concerns of the users.).

To cope with this challenge, some recent efforts have been made to develop effective network embedding models dedicated for jointly modeling structural and textual information, which have received considerable effects. For instance, TADW [22] is the first work that incorporates textual features into network embeddings through matrix factorization. Sun et al. [23] proposed a general framework (CENE) to jointly learn the information of network structure and content. They treated the content information as a special type of node and constructs an augmented network involving node–node links and node-content links. Tu et al. [24] presented a context-aware network embedding model (CANE). They assumed that a node shows different aspects when interacting with different context and applied a mutual attention mechanism to precisely model the semantic relationships between nodes. However, there are two major limitations of existing embedding methods for textual information networks. The first one lies in that text-attribute relationship between vertices is easily subject to the topological structure of the network under current proposed schemes. As a result, existing methods cannot comprehensively and independently learn the deep semantic relationship of text attributes between nodes. In reality, if two nodes are not linked together in an information network, it does not necessarily indicate they are dissimilar. Taking the social website as an example, although there is no direct interaction between two users, their hobbies or research orientations are very similar. It is simply because they have no chance to know each other, thus they should be forced closer in the embedding space. Analogically, two nodes that are just connected together cannot ensure they are similar. The second limitation arises from the scalability problem. As we know, matrix approximation/decomposition methods have shown great potential in the many research fields including network embeddings [22,25–27], but their scalability is still a main bottleneck, which limits their better applications and gives great opportunity to random walk based techniques. Random walks benefit from their low computational complexity in exploiting local topology information but are inefficient even infeasible to manipulate node attributes. On the other hand, at present the most commonly used matrix decomposition methods, such as singular value decomposition (SVD) [28], principal component analysis (PCA) [29], nonnegative matrix decomposition [30], can accurately capture global information of a network. However, when the scale of the network reaches millions or even higher, the computational complexity of conventional matrix decomposition methods is very high. Hence, there raises an urgent demand for improving the scalability of current information network embedding methods.

Motivated by the above observations, we propose an *Enhanced Textual Information Network Embedding* (*ETINE*) method to efficiently integrate both network structure and text attributes of nodes. ETINE is a unified framework which learn network embeddings with not only global structure feature but also deep semantic relationship between nodes. More specifically, two types of proximity matrices are properly constructed through probabilistic sampling method to measure multiple proximities of nodes – K orders of structural proximity matrices that characterizes global structural similarity and a textual proximity matrix that accurately describes textual similarity relationships. Then, we define the structure-based and text-based loss functions for learning network embeddings with structural and textual proximity information, and prove that optimizing them involves a matrix approximation problem. Consequently, we introduce a novel efficient low-rank matrix approximation method, called *randomized singular value decomposition* (RSVD) [31], to decompose our loss functions. RSVD has been successfully applied in many research fields, e.g., hyperspectral image processing [32], determinant point processes [33], etc. To the best of our knowledge, we are the first to utilize RSVD method to solve the optimization problem of network embedding.

In summary, the main contributions of this paper are as follows:

- We present an efficient and scalable textual information network embedding framework (ETINE) for learning network embeddings with not only global structural information but also deep semantic relationship between nodes in a network.
- Randomized singular value decomposition (RSVD) is firstly introduced and extended to solve our proposed optimization problem. Benefiting from probabilistic analysis, RSVD method can improve the accuracy of the model with lower complexity than conventional low-rank approximations, such as SVD and PCA.
- Extensive experiments are conducted on four representative real-world networks. The results demonstrate that the superior performance of ETINE on the tasks of network reconstruction, node clustering and multi-class node classification. More specifically, our model outperforms baselines by 40.18% (*NMI*) in node clustering, 17.83% (*Macro-F1*) and

Table 1
A summary of notations.

Notations	Descriptions
$G = (V, E, T)$	Information network G with node set V, edge set E and text set T
K, k	Maximal transition step K and the specific transition step k
w, $\overrightarrow{w}$	Current node during the transitional process and the vector of w
c, $\overrightarrow{c}$	Context node during the transitional process and the vector of c
n, $\overrightarrow{n}$	Randomly sampled "negative" context node and the vector of n
A, S, S^k	Adjacency matrix, 1-order and k-order structural proximity matrices
T	1-order textual proximity matrix
D_s, D_t	Structure-based and text-based Degree diagonal matrix of G
Y_s^k, Y_t	The structural and textual log probabilistic matrices
X_s^k, X_t	The structural and textual positive log probabilistic matrices
W_s^k, W_t	The structural and textual representation matrices of current node
C_s^k, C_t	The structural and textual representation matrices of context node
W	Final node representations of G
C	Representations of context nodes
d	Dimension size of the embedding space
β	Log shifted factor
λ	Number of negative samples

13.51% (*Micro-F1*) in multi-class node classification when the training ratio is 90%.

The remainder of this paper is organized as follows. In Section 2, the problem definition and related works are given. The proposed model is illustrated in detail in Section 3. In Section 4, the experimental settings and results are provided. We conclude this paper and discuss the future work in Section 5.

2. Background

In this section, we first give a summary of the notations that are used in our model, followed by the formal definitions of the problem. Afterwards, we review some works related to network embedding.

2.1. Notations and definitions

The detailed list of common notations used throughout the paper can be found in Table 1.

Definition 1 (*Information Network*)**.** *An information network is generally denoted as* $G = (V, E, T)$*, where* $V = \{v_1, v_2, \ldots, v_N\}$ *denotes the vertices;* $E \subseteq V \times V$ *denotes edges between vertices, and* T *denotes the text attributes of nodes. Each edge* $e_{i,j} \in E = \{(v_i, v_j) | v_i, v_j \in V\}$ *between* v_i *and* v_j *can be associated with a weight* $\varphi_{i,j}$ *representing the strength of the connection. Each* $t_i \in T$ *representing the text attributes of* v_i *is a word sequence* $(\tau_1, \tau_2, \ldots, \tau_{n_i})$*.*

We assume the information network is directed. If there is an undirected network, an edge can be identified as two oppositely directed edges with a same weight. The adjacency matrix A is used to characterize the neighboring relationships between nodes. If there is an edge from v_i to v_j, we have $A_{i,j} = \varphi_{i,j}$ for weighted graph, $A_{i,j} = 1$ for unweighted graph; otherwise, $A_{i,j} = 0$. Although in some cases, the weights of edges can be negative, we only focus on non-negative weights.

Definition 2 (*Structural Proximity*)**.** Structural proximity of nodes is induced by links. Given nodes v_i and v_j, if there exists a link between them, it indicates the first-order proximity. On the other hand, if v_i belongs to the context of v_j, it indicates the higher-order proximity.

There are many effective ways to generate contexts for nodes, e.g., constructing relationship/proximity matrix, performing random walks, etc. In our method, we apply the method similar to [25], which models global structural properties of the network through constructing K orders of structural proximity matrices. Global structural properties not only enable extract the long-distance relationship between two different nodes, but also consider different connectivity structure of the network in terms of distinct transition steps.

Definition 3 (*Textual Proximity*)**.** Textual proximity of nodes is evidenced by text attributes. Given two nodes v_i and v_j, the text attribute intersection of t_i and t_j indicates the textual proximity of v_i and v_j.

By considering the textual proximity, the homophily effect of text attribute can be modeled as nodes with similar textual information will be placed close to each other in the embedding space. It is worth noting that since there is no higher-order proximity information in textual relationships, we only need to consider the direct textual proximity.

Definition 4 (*Network Embedding*)**.** Given a network denoted as $G = (V, E, T)$, Network embedding attempts to obtain a low-dimensional continuous and non-sparse vector $W \in \mathbb{R}^d$ for every node $v \in V$, where $d \ll |V|$.

For textual information networks, because the topological structure and text attributes of nodes provide different sources of network information, the goal of our proposed ETINE is to maintain both of them to learn more comprehensive network embeddings.

2.2. Related works

As mentioned before, the aim of network embedding is to represent the nodes of a network into a low-dimensional embedding space, retaining the network information as much as possible. The difference between different network embedding methods lies in how they define the network information that needs to be retained and how to store them in the embedding space. Next, we will introduce some related network embedding methods and how they can be used to solve the problem of network embedding. After that, we summarize current network embedding approaches that account for extra textual information.

2.2.1. Plain network embeddings

In recent years, many researchers have paid a lot of attention in the field of network embedding, aiming to design scalable methods that can effectively retain network information. These methods can be divided into three categories: *matrix-decomposition based methods*, *random-walk based methods* and

deep-learning based methods. (1) The methods based on matrix decomposition focus on capturing the interactions between network nodes in terms of matrix, and employs matrix decomposition based method to learn the low-rank representation of nodes. For instance, in order to preserve the network topology information, one can construct different types of matrices, such as *k*-order probability transition matrix, modularity matrix, context-aware matrix and so on. In GraRep, Cao et al. [25] employed singular value decomposition (SVD) to construct the *K*-order probability transition matrix which describes the *k*-order proximity structure between nodes to learn the global structure characteristics of the network. MMDW [26] is a semi-supervised model that aims to learn discriminative representations of nodes. In M-NMF, Wang et al. [27] used nonnegative matrix decomposition [30] to preserve the micro-structure (including the first-order and second-order similarity) and macroscopical community structure. (2) The random-walk based methods simulate natural language sentences by sampling the random walk sequence to capture the structural proximity between nodes in the network. Random walk is usually used to extract network structure, including structure-role similarity and community similarity. DeepWalk [34] is a pioneer work using random walk technique to learn node representations. Node2vec [20] further uses biased random walk strategy to obtain more flexible context structure. Metapath2vec [35] studies representation learning in heterogeneous networks [36]. It formalizes heterogeneous neighborhood of nodes by random walk process, and then uses heterogeneous SkipGram [37] model to learn node representations. [38] proposed a deep network embedding framework (DNE), and designed a degree weighted random walk technique to extract the global context information of the network. (3) The methods based on deep learning mainly adopt deep learning technology to capture the deep structural features of the network and learn highly nonlinear node representations. For example, SDNE [11] proposes a semi-supervised deep model, which uses autoencoder mechanism to capture highly nonlinear network structure; DNGR [17] embeds random surfing model into network structure, and uses stacked denoising autoencoder (DAE) to learn efficient node representations; GraphGAN [39] utilizes the framework of generative adversarial networks (GAN) to accurately model node connectivity probability. Then, Lee et al. [40] proposed a signed network embedding method, motivated by the success of generative adversarial networks. For dynamic networks, Yu et al. [41] proposed to tackle the dilemma via matrix perturbation. This method implements a low-rank transformation on the normalized Laplacian matrix and derives the embeddings via SVD. All the above methods only consider the topology of networks, without considering the rich attribute information of the node itself accompanied with nodes in real-world networks.

2.2.2. Textual information network embeddings

To address this issue, some recent efforts have explored to integrate heterogeneous information (e.g., text attributes) into conventional network embedding methods. For example, TADW is the first work that combines structure and textual information for learning network embeddings. Yang et al. [22] presented text-associated Deep Walk (TADW) to improve matrix factorization based DeepWalk with text information. However, contents in TADW are not explicitly modeled but just simply integrated as unordered text features. To address this issue, Sun et al. [23] treated text content as a special kind of node and proposed a context-enhanced network embedding model (CENE) by utilizing both structural and textual information to learn network embeddings. In Tri-DNR, Pan et al. [42] proposed to leverage information from three parties: node structure, node content and/or node labels and learn separate embeddings for them. The learned embeddings were linearly combined together through sum operations in an iterative manner. More recently, Tu et al. [24] proposed a context-aware network embedding (CANE) model that learns context-aware embeddings for nodes with mutual attention mechanism to precisely model the semantic relationships between nodes. Liao et al. [1] proposed a general neural network framework (ASNE), by capturing the network structure similarity and the node attribute correlation degree to learn the vector representation of nodes. Motivated by the strong representation and generalization ability of deep learning models, ASNE uses a multilayer neural network to model the complex relationship between network structure information and node attributes. However, the above approaches are incapable of capturing global textual relationships between nodes, thus may result in less informative embeddings. That is, textual properties are easily subject to the topological structure of the network. As a result, the learned embeddings cannot represent the pure textual relationships between nodes. On the contrary, our ETINE considers deep semantic relationship independently, without getting affected by the original topological structure of the network. After that, we combine the learned structural embeddings and textual embeddings to obtain the final node representations. Liao et al. [1] proposed ASNE which can utilize the rich attribute information and learn the representations for social actors by preserving both the structural proximity and attribute proximity. Gao et al. [9] designed a margin-based random walk to incorporate the community information. This method integrates textual semantics into the representations by using the topic model. Cheng et al. [43] focused on the dynamic textual network and designed a diffusion to model multiple hops between nodes, attempting to automatically learn the importance of global and local structure for the embeddings. Huang et al. [44] focused on the bipartile attributed network and proposed a model to integrate the inter-partition proximity and intra-partition proximity. Cui et al. [45] proposed an adaptive graph encoder for attributed graph embedding. This method applies a designed filter and an adaptive encoder to strengthen the features for node embeddings. Chen et al. [46] utilized an attention mechanism to adaptively aggregate the information based on the neighbors' importance, which can handle well with the sparse attributed network embeddings. Ali et al. [47] proposed a method which can learn researchers' and papers' dynamics by encoding information from six networks into a latent space to overcome the issue of cold start papers and data sparsity. Zhang et al. [48] proposed to integrate the information from diverse relations by considering the relative importance of these relations.

3. Enhanced Textual Information Network Embedding (ETINE)

In real-world textual information networks, network proximities are induced by direct interactions between nodes, accompanied with rich text attribute information. In order to characterize the closeness of the relationship between nodes, we propose to model both the structural and textual proximities between nodes in networks. Probabilistic sampling is employed in the process of constructing the structural and textural proximity matrices. It is more efficient than the uniform sampling method used in DeepWalk [34] and Tri-DNR [42] models, because uniform sampling methods contain many parameters that need to be adjusted, such as the length of sampling sequence, sampling frequency, etc. After that, we give the definitions of our loss functions which model global structural feature and deep semantic relationship between nodes, respectively. Finally, we introduce and extend an efficient matrix approximation method (RSVD) to optimization the proposed loss functions. In the following, we will detail our ETINE model. Intuitively, the overall framework of our ETINE is shown in Fig. 2.

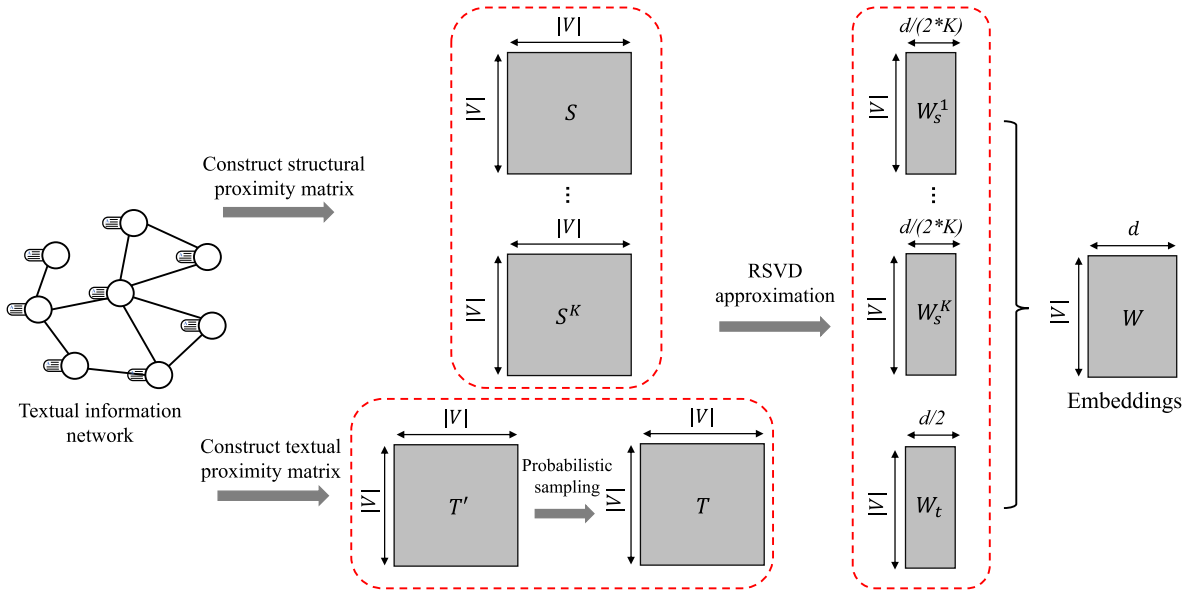


Fig. 2. Overall framework of ETINE.

3.1. Loss functions on network

Information networks belong to a special class of networks, in which the formation of proximities involves not only the self-organizing network process, but also the attribute-based process [1]. Nodes in many information networks are sparsely connected but each node usually is accompanied with rich text information. As thus, the text attributes can be used to complement the weaknesses of the structure part. In this section, we elaborate the modeling process of the structural and textual proximities, which is critical for learning comprehensive embeddings of networks.

3.1.1. Structural part

At present, the most commonly used methods for feature extraction are random walks. However, random walks always arise the problem of slow and expensive sampling process, caused by many hyper-parameters to tune. In this work, motivated by GraRep [25], we use transition probabilities to measure relationship between nodes. Given an information network $G = (V, E, D)$, represented by the adjacency matrix $A = \{a_{ij}\}$. With the adjacency matrix, the corresponding diagonal matrix D_s can be computed as follows:

$$(D_s)_{ij} = \begin{cases} \sum_p A_{ip}, & i = j \\ 0, & i \neq j \end{cases} \tag{1}$$

Here we introduce a *(1-step) probability transition matrix* $S = D_s^{-1}A$ to capture the transitions from one node to another, where each $s_{ij} \in S$ denotes the probability of a transition from v_i to v_j within one step. In order to learn global representation of a network, we capture k-step relational information between different nodes in a network, with different values of k. Based on the definition of 1-step probability transition matrix S, the k-step probability transition matrix S^k that represents the k-step structure transition probabilities can be computed as follows:

$$S^k = \underbrace{S \cdots S}_{k} \tag{2}$$

We adopt the probability transition matrix S^k with different k as the structural proximity matrices with different orders, where $s^k_{w,c}$ represents k-step structure transition probability from node w to c. More formally, we have:

$$p_k(c|w) = S^k_{w,c}. \tag{3}$$

Inspired by Skip-Gram proposed by Mikolov et al. [37], we use *noise contrast estimation (NCE)* proposed by Gutmann and Hyvarinen [49] to define the structure-based objective $L_s(w)$ as follows:

$$L_s(w) = \sum_{k=1}^{K} \sum_{w \in V} L_{s^k}(w), \tag{4}$$

where each $L_{s^k}(w)$ is defined as:

$$L_{s^k}(w) = \sum_{c \in V} p_k(c|w) \log \sigma\left(\vec{w} \cdot \vec{c}\right) + \lambda \cdot \mathbb{E}_{n \sim p_k(V)}\left[\log \sigma\left(-\vec{w} \cdot \vec{n}\right)\right], \tag{5}$$

where $p_k(c|w)$ represents the k-step structural transition probability from node w to c (Here w and c represent the current node and context node in the transition process, respectively), $\sigma(x) = \left(1 + e^{-x}\right)^{-1}$ is sigmoid function, λ denotes the number of negative samples, and $\mathbb{E}_{n \sim p_k(V)}[\cdot]$ denotes the expectation when the context node n follows the distribution of $p_k(V)$, where n is the "negative" context randomly sampled from the network. Therein, the aim of the first term is to model the probability that (w, c) pair comes from the network, while the aim of the second term is to model the probability of the negative samples not coming from the network. The second term can be further expanded as:

$$\mathbb{E}_{n \sim p_k(V)}\left[\log \sigma\left(-\vec{w} \cdot \vec{n}\right)\right] = p_k(c) \cdot \log \sigma\left(-\vec{w} \cdot \vec{c}\right) + \sum_{n \in V \setminus \{c\}} p_k(n) \cdot \log \sigma\left(-\vec{w} \cdot \vec{n}\right). \tag{6}$$

According to Eqs. (5) and (6), we define a local loss over a node pair (w, c) as:

$$L_{s^k}(w, c) = p_k(c|w) \cdot \log \sigma(\vec{w} \cdot \vec{c}) + \lambda \cdot p_k(c) \cdot \log \sigma(-\vec{w} \cdot \vec{c}). \tag{7}$$

In fact, the transition probabilities will converge to a fixed value when the transition step k is large enough. The distribution $p_k(c)$

can be calculated as:

$$p_k(c) = \sum_u q(u)p_k(c|u) = \frac{1}{N} \cdot \sum_u S^k_{u,c}, \tag{8}$$

where N is the total number of nodes in the network. $q(u)$ means the probability of picking node u as the first node in the transition process. Here we assume $q(u)$ follows a uniform distribution, i.e., $q(u) = 1/N$. This leads to:

$$L_{s^k}(w, c) = S^k_{w,c} \cdot \log\sigma\left(\overrightarrow{w} \cdot \overrightarrow{c}\right) + \frac{\lambda}{N}\sum_u S^k_{u,c} \cdot \log\sigma\left(-\overrightarrow{w} \cdot \overrightarrow{c}\right). \tag{9}$$

In order to optimize (9), let $y = \overrightarrow{w} \cdot \overrightarrow{c}$ and set partial derivative term $\frac{\partial L_{sk}}{\partial y}$ to zero. We can derive the following formula:

$$y = \overrightarrow{w} \cdot \overrightarrow{c} = \log\left(\frac{S^k_{w,c}}{\sum_u S^k_{u,c}}\right) - \log(\beta) \tag{10}$$

where $\beta = \lambda/N$.

This certifies that the matrix Y^k_s essentially can be factorized into two matrices W^k_s and C^k_s, where each row of W^k_s and C^k_s represents a vector of the node w and c respectively [50], and Y^k_s is formulated as:

$$Y^k_{s_{i,j}} = W^k_{s_i} \cdot C^k_{s_j} = \log\left(\frac{S^k_{i,j}}{\sum_m S^k_{m,j}}\right) - \log(\beta) \tag{11}$$

where $Y^k_{s_{i,j}}$ is the element of ith row and jth column of the matrix Y^k_s, $W^k_{s_i}$ is the vector of ith row of the matrix W^k_s, and $C^k_{s_i}$ is the vector of jth row of the matrix C^k_s. In this way, we prove that the defined loss function can be optimized by matrix approximation methods.

3.1.2. Textual part

Text attributes are the text-based information (i.e., word sequence) attached to the node, which cannot be used directly. Thus, we need to learn text embeddings at first. On the other hand, as we discussed before, textual relationship should be modeled independently without being affected by the original topology of the network. Yang et al. [22] used the term frequency–inverse document frequency (TFIDF) matrix to extract textual information, which simply measures the importance of each word in the sentence by word frequency, without considering the order of word sequence and deep semantic information. Instead of using bag-of-words features that ignore the order of words and syntax of sentence, this paper investigates three different neural networks for text modeling. Their model structures are illustrated in Fig. 3.

Doc2vec (D2V) [51]. This approach is motivated by word2vec, to compute text embeddings from text attributes with variable length. Here, we employ the Distributed Bag of Words version of Paragraph Vector (PV-DBOW), in which the text embedding is served as the input to predict the probability of randomly sampled words in the output text.

Attention-based Convolutional Neural Network (ATT-CNN) [52]. Given a word sequence of one node, this approach can obtain the vector through the convolution layer and the max-pooling layer. Note that the attention mechanism is applied before the pooling layer [24], which enables the vectors of node pair in an edge to influence each other.

Bidirectional Recurrent Neural Network (BiRNN) [53]. Here we use BiRNN, the cells of which are the Long Short-term Memory (LSTM) [54] cells, to capture long-term dependencies of sentences in both directions. These two hidden state vectors obtained from forward-directional RNN (fRNN) and backward-directional RNN (bRNN) are then concatenated to form a whole vector.

After text modeling, motivated by the adjacency matrix of a network that characterizes the structural neighboring relationships between nodes, we describe the textual similarity measured by the cosine similarity metric, which is computed as:

$$T'_{i,j} = \cos(\overrightarrow{h_i}, \overrightarrow{h_j}) = \frac{\overrightarrow{h_i} \cdot \overrightarrow{h_j}}{\left|\overrightarrow{h_i}\right| \cdot \left|\overrightarrow{h_j}\right|}, \tag{12}$$

where $\overrightarrow{h_i}$ and $\overrightarrow{h_j}$ represent text embeddings of node v_i and v_j, respectively. According to the definition of proximity matrix, we use probability sampling method to measure the relationship between text attributes. Since there is no higher-order proximity information between text attributes, we define the textual proximity matrix T as follows:

$$T = D_t^{-1} \cdot T' \tag{13}$$

where D_t is the textual diagonal(i.e., degree) matrix as follows:

$$(D_t)_{ij} = \begin{cases} \sum_m T'_{i,m}, & i = j \\ 0, & i \neq j \end{cases} \tag{14}$$

Because the textual proximity matrix T is also constructed based on the probabilistic sampling method, to be consistent with $L_s(w)$, we define the text-based objective $L_t(w)$ as follows:

$$L_t(w) = \sum_{c \in V} T_{w,c} \log\sigma\left(\overrightarrow{w} \cdot \overrightarrow{c}\right) + \lambda \cdot \mathbb{E}_{n \sim p_t(V)}\left[\log\sigma\left(-\overrightarrow{w} \cdot \overrightarrow{n}\right)\right] \tag{15}$$

Here, $T_{w,c}$ is the element of wth row and c-column of the matrix T and describes the textual transition probability from node w to c. The second term uses the same expansion method as Eq. (6) except that the "negative" context node n follows $p_t(V)$, where $p_t(c) = \frac{1}{N} \cdot \sum_u T_{u,c}$. The definitions of the remaining parameters are similar to Section 3.1.1, including the optimization process. The matrix Y_t can also be factorized into two matrices W_t and C_t as:

$$Y_{t_{i,j}} = W_{t_i} \cdot C_{t_j} = \log\left(\frac{T_{i,j}}{\sum_m T_{m,j}}\right) - \log(\beta) \tag{16}$$

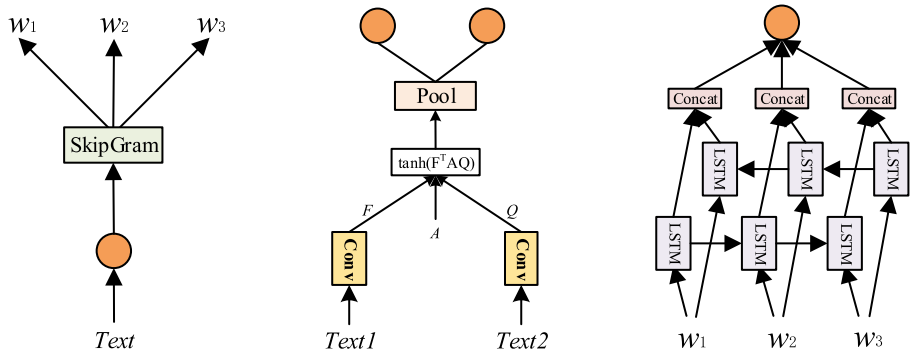
where each row of W_t and C_t represents a vector of the node w and c respectively, $Y_{t_{i,j}}$ is the element of ith row and jth column of the matrix Y_t, W_{t_i} is the vector of ith row of the matrix W_t, and C_{t_i} is the vector of jth row of the matrix C_t.

3.2. Optimization with matrix approximation

How to effectively retain and learn both the topological structure and text attributes of nodes is the focus of this paper. As shown in Eqs. (4) and (15), the optimization of ETINE's objective functions is proved to be a matrix approximation problem. Since the high computational complexity of the most commonly used methods, such as SVD [22,25], PCA [29], we introduce a more efficient randomized singular value decomposition method, namely RSVD, to solve the matrix approximation problem.

3.2.1. RSVD method

The process of RSVD is divided into two stages [31]. In stage one, a low-dimensional subspace is constructed to capture the most of the information of the original matrix X using random sampling. In stage two, the reduced matrix is then decomposed using SVD. The detailed process is given in Algorithm 1. A flowchart shown in Fig. 4 to emphasize on the working steps of the proposed method.



(a) Doc2vec (b) ATT-CNN (c) BiRNN

Fig. 3. Three deep learning based models for text modeling, in which orange circles represent text embeddings.

Algorithm 1 RSVD Method

Input: $m \times n$ matrix X, and number of singular values r.
Output: Left singular vectors U, singular values matrix Σ, and right singular vectors V.
Stage 1:
1. Generate a $n \times r$ Gaussian random matrix Ω.
2. Compute $H = X\Omega$.
3. Construct a matrix Q via QR factorization of H, where the columns of Q form an orthogonal basis for the range of H.
Stage 2:
4. Compute $B = Q^T X$.
5. Calculate SVD factorization of the reduced matrix B: $B = \widetilde{U}\Sigma V^T$.
6. Set $U = Q\widetilde{U}$.

Note that, r is much smaller than m and n for unnecessary storage. As Algorithm 1 shows, RSVD only requires $O\left(mn\log r + (m+n)r^2\right)$ flops, while SVD requires $O\left(mn \cdot \min\{m,n\}\right)$ flops, which demonstrates the high-efficiency of RSVD method.

3.2.2. Approximation with RSVD method

As introduced in [55], to reduce the impact of noise, all negative elements in Y_s^k and Y_t are replaced by 0 to present positive log probabilistic matrices, where:

$$X_{t_{i,j}} = \max\left(Y_{t_{i,j}}, 0\right), \quad X_{s_{i,j}}^k = \max\left(Y_{s_{i,j}}^k, 0\right). \tag{17}$$

In order to obtain the optimal rank-r approximation, we perform RSVD on X_s^k and X_t via Algorithm 1. Formally, we first approximate:

$$X_s^k \approx X_{s_{d'}}^k = U_{s_{d'}}^k \Sigma_{s_{d'}}^k \left(V_{s_{d'}}^k\right)^T \tag{18}$$

where the number of singular values r is denoted as $d' = d/(2 \cdot K)$, and $\Sigma_{s_{d'}}^k$ is the diagonal matrix which is consisted of the top d' singular values, $U_{s_{d'}}^k$ and $V_{s_{d'}}^k$ are the corresponding columns of the matrix U_s^k and V_s^k. Therefore, X_s^k can be factorized as:

$$X_s^k \approx X_{s_{d'}}^k = W_s^k \cdot C_s^k \tag{19}$$

where

$$W_s^k = U_{s_{d'}}^k \cdot \sqrt{\left(\Sigma_{s_{d'}}^k\right)}, \quad C_s^k = V_{s_{d'}}^k \cdot \sqrt{\left(\Sigma_{s_{d'}}^k\right)}. \tag{20}$$

Similarly, for the matrix X_t, the number of singular values r is $d/2$, and X_t can be calculated as:

$$X_t \approx W_{t_{d/2}} = W_t \cdot C_t \tag{21}$$

where

$$W_t = U_{t_{d/2}} \cdot \sqrt{\left(\Sigma_{t_{d/2}}\right)}, C_t = V_{t_{d/2}} \cdot \sqrt{\left(\Sigma_{t_{d/2}}\right)}. \tag{22}$$

In Eqs. (20) and (22), each row of W_s^k and W_t represents the structure-based and text-based representations of the current node, respectively, and each row of C_s^k and C_t represents the structure-based and the text-based representations of the context node separately. The final node representation W is concatenated with W_s and W_t as follows:

$$W = [W_s, W_t] = \left[W_s^1, \ldots, W_s^K, W_t\right], \tag{23}$$

where $[\cdot]$ denotes the concatenation operation. The pseudo code of the proposed ETINE is shown in Algorithm 2. Firstly, according to the topological structure and the text attributes of the nodes, the structural proximity matrix and the textual proximity matrix are constructed, respectively. Then, the negative number of the proximity matrix is replaced by 0, and the RSVD method is used to reduce the dimension of the matrix. The structure-based embedding matrix and text-based embedding matrix are concatenated as the final embedding matrix of the network. In this method, the matrix factorization is most computationally demanding. Thus, the overall computational complexity has the same averaged upper bound with RSVD, namely, $O(n^2(log(r) + r^2) \cdot K)$, where n is the number of nodes, r is the estimated number of singular values, and K is the maximum-step of probability transition. In fact, since each k-step transition probability matrix is dependent mutually, the matrix factorization of all matrices can be performed in parallel, leading to an upper bound of $O(n^2(log(r) + r^2))$.

4. Experiments

In order to evaluate the effectiveness of the proposed ETINE, we conduct comparative experiments on four real-world textual information networks, including three citation networks, and an online Q&A website. The experimental applications involve multi-class node classification, node clustering, network reconstruction and parameter sensitivity analysis.

4.1. Datasets

The statistics of the four datasets are summarized in Table 2.

Algorithm 2 ETINE Model

Input: $G = (V, E, D)$, Adjacency matrix A, maximal transition step K, shifted parameter β, dimension size d.

Output: Node representation W.

1. Calculate proximity matrices S^k and T according to (2) and (13).

2. Calculate log probabilistic matrices Y_s^k and Y_t via (11) and (16).

3. Replace negative elements of Y_s^k and Y_t with 0 to get the positive log probabilistic matrices X_s^k and X_t.

4. Perform RSVD with X_s^k and X_t respectively, and obtain the global structure-based representation W_s and the text-based representation W_t via (20) and (22).

5. Concatenate W_s with W_t to obtain the final node representation W according to (23).

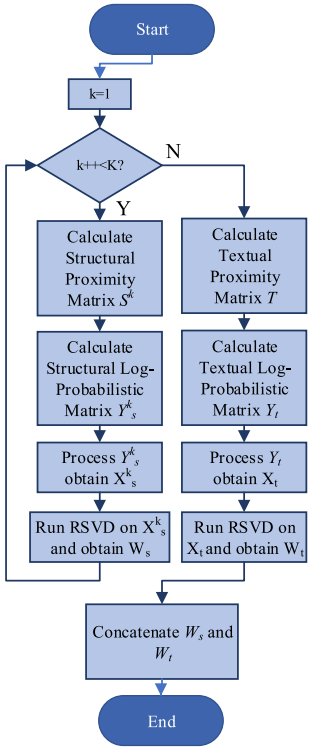


Fig. 4. The flowchart of ETINE.

- **Cora** [56] is a typical citation network including 2277 machine learning papers and 5214 links, which can be divided into 7 categories. The linkages between papers indicate the citing relationships of them.
- **CiteSeer-M6** [42] is the subset of a citation network CiteSeer-M10, which is composed of six classes: Biology, Computer Science, Financial Economics, Industrial Engineering, Physics and Social Science. There are 4398 publications and 5636 relations between the publications in CiteSeer-M6. Specifically, the text attribute of each node is the title of the paper.
- **DBLP** [3] is also a citation network, which contains 60 744 nodes and 52 890 links. Similar to CiteSeer-M6 dataset, the text attributes of nodes are the corresponding titles of the papers. DBLP network dataset can be divided into 4 groups in terms of different research orientations. The detailed contents are shown in Table 3.
- **Zhihu** [24] is a Chinese online Q&A website, whose goal is constructing an online community for users to query and answer questions. In this paper, we randomly crawl the descriptions of 10 000 active users' relevant topics as the text information of nodes.

Table 2
Statistics of datasets.

Datasets	Cora	CiteSeer-M6	DBLP	Zhihu
#Vertices	2277	4398	60 744	10 000
#Edges	5214	5636	52 890	43 894
#Labels	7	6	4	–

4.2. Evaluation metrics

In the experiments, *Micro-F1* and *Macro-F1* are used as evaluation metrics for multi-class node classification and parameter sensitivity analysis, where *F1* value is a criteria to evaluate binary classifiers. Given a label m, we denote True Positive, False Positive and False Negative as *TP*, *FP* and *FN* separately. M is assumed to be the label set. Then the definitions of *Micro-F1* and *Macro-F1* are described below:

Micro-F1 is a criterion that calculates global metric by counting the total *TP(m)*, *FP(m)*, *FN(m)*, which is defined as:

$$Micro - F1 = 2\frac{Pr_{mic} \times Re_{mic}}{Pr_{mic} + Re_{mic}} \tag{24}$$

where Pr_{mic} indicates precision for micro, and Re_{mic} represents recall for micro:

$$Pr_{mic} = \frac{\sum_{m\in M} TP(t)}{\sum_{m\in M} TP(t) + FP(t)}, Re_{mic} = \frac{\sum_{m\in M} TP(t)}{\sum_{m\in M} TP(t) + FN(t)}. \tag{25}$$

Macro-F1 is a criterion that calculates metric for every label and obtains their unweighted mean values, which is calculated as:

$$Macro - F1 = 2\frac{Pr_{mac} \times Re_{mac}}{Pr_{mac} + Re_{mac}} \tag{26}$$

where Pr_{mac} indicates average precision for macro and Re_{mac} represents average recall for macro:

$$Pr_{mac} = \frac{\sum_{m\in M} TP(m)/(TP(m) + FP(m))}{|M|},$$
$$Re_{mac} = \frac{\sum_{m\in M} TP(m)/(TP(m) + FN(m))}{|M|}. \tag{27}$$

For the task of node clustering, we introduce the *normalized mutual information* (*NMI*) [57] to assess the effect. *NMI* is a criterion that measures the closeness between ground truth of the real label set O and the predicted label set L:

$$NMI(O, L) = \frac{-2 \times \sum_{i=1}^{|O|}\sum_{j=1}^{|L|} F_{ij} \log\left(F_{ij}N/F_{i\cdot}F_{\cdot j}\right)}{\sum_{i=1}^{|O|} F_{i\cdot} \log\left(F_{i\cdot}/N\right) + \sum_{j=1}^{|L|} F_{\cdot j} \log\left(F_{\cdot j}/N\right)}, \tag{28}$$

where F_{ij} denotes the element of a confusion matrix F, which represents the number of nodes in class i of the set O that have also been found in class j of the set P. And $F_{i\cdot}$ and $F_{\cdot j}$ represent the sum of elements over row i and column j of F, respectively. The *NMI* value ranges from 0 to 1.

For the task of network reconstruction, we adopt *precision@h* to evaluate the performance [11]. *precision@h* calculates the prediction accuracy of the top h edges as follows:

$$precision@h\,(i) = \frac{\left|\left\{j \middle| i, j \in V, index\,(j) \leqslant h, \delta(i, j) = 1\right\}\right|}{h} \tag{29}$$

Table 3
Groups of DBLP dataset.

Conference	Group
SIGMOD, ICDE, VLDB, EDBT, PODS, ICDT, DASFAA, SSDBM, CIKM	Database
KDD, ICDM, SDM, PKDD,PAKDD	Data Mining
IJCAI, AAAI, NIPS, ICML, ECML, ACML, IJCNN, UAI,ECAI, COLT, ACL, KR	Artificial Intelligent
CVPR, ICCV, ECCV, ACCV, MM, ICPR, ICIP, ICME	Computer Vision

Table 4
Micro-F1 and Macro-F1 on Cora dataset.

Metric	Model	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1	DeepWalk	76.34	79.39	80.59	81.61	82.53	83.15	83.51	83.54	83.42
	LINE	77.62	79.54	80.54	80.94	81.29	81.53	81.52	81.40	83.02
	GraRep	75.88	77.58	78.23	78.88	79.07	79.41	79.77	80.45	79.38
	Node2vec	75.59	78.08	79.70	80.38	80.99	81.44	81.73	82.08	81.94
	TADW	73.59	79.62	82.07	83.29	84.16	84.47	85.00	84.88	85.45
	CANE	78.94	81.47	82.25	83.21	83.56	83.94	83.96	84.18	84.91
	COANE	79.12	81.90	82.70	84.23	84.61	85.08	85.91	**86.59**	86.82
	ETINE-D2V	78.80	81.30	82.93	83.84	84.36	85.19	85.39	86.14	86.93
	ETINE-CNN	**80.06**	**83.14**	**84.17**	**84.88**	**85.42**	**85.65**	**86.08**	86.37	**86.98**
	ETINE-BiRNN	78.14	81.20	82.64	82.98	83.57	84.19	84.59	85.50	85.24
	ETINE-Bert	77.26	79.58	80.91	81.72	82.42	82.82	83.04	84.24	85.45
Macro-F1	DeepWalk	75.74	78.99	80.20	81.09	81.84	82.51	82.68	82.50	81.83
	LINE	76.98	79.17	80.14	80.51	80.93	81.12	81.04	80.77	82.55
	GraRep	75.15	76.87	77.49	78.04	78.19	78.40	78.88	79.38	78.94
	Node2vec	75.38	77.92	79.59	80.09	80.68	81.25	81.46	81.85	81.17
	TADW	72.09	78.89	81.37	82.56	83.47	83.82	84.09	84.09	84.60
	CANE	76.21	79.88	80.94	82.21	82.62	83.00	83.01	83.06	83.81
	COANE	77.50	80.67	81.34	82.78	83.41	83.62	84.71	85.40	85.51
	ETINE-D2V	78.32	80.52	82.10	83.19	83.56	83.91	84.28	85.31	85.75
	ETINE-CNN	**79.04**	**82.54**	**83.65**	**84.27**	**84.73**	**85.07**	**85.62**	**85.73**	**85.81**
	ETINE-BiRNN	77.42	80.55	82.13	82.42	82.93	83.52	83.84	84.86	84.62
	ETINE-Bert	76.49	78.97	80.34	81.72	82.13	83.24	83.67	84.56	84.82

where $index(j)$ indicates the sorting index of v_j, and $\delta(i,j) = 1$ denotes that there is a link between v_i and v_j.

4.3. Baselines

In this paper, we employ the following methods as our baselines, including four structure-only methods and two structure-and-text methods.

- **DeepWalk** [34] applies SkipGram model with hierarchical softmax over random walks sampled from the network to learn vector embedding for each node.
- **LINE** [58] learns two embedding vectors for each node by maintaining the first-order and second-order proximities of the network. Then, they are concatenated as the final embedding for a node.
- **Node2vec** [20] introduces biased random walks by using both breadth-first sampling and depth-first sampling. Then, the SkipGram model is applied to learn node representations. It has two critical hyper-parameters p and q that guide the walks.
- **GraRep** [25] defines k-step loss functions for extracting the different k-step local relational information and optimizes each model with matrix decomposition. The global embedding for each node is constructed by combining different embeddings learned from different models for further processing.
- **TADW** [22] encodes text information of nodes into network representation learning under the scheme of matrix factorization. In their model, different types of information are combined by jointly modeling them via matrix factorization.
- **CANE** [24] employs a mutual attention mechanism to learn various context-aware embeddings for a node according to their interacted neighbors. It also learns two types of embeddings for a node, i.e., structure-based embedding and text-based embedding. The final embedding for each node is obtained by simply concatenating them.
- **COANE** [9] contains a margin-based random walk to incorporate the community information. This method integrates textual semantics into the representations by using the topic model.

4.4. Models and configurations

We evaluate our proposed model ETINE for tasks of network reconstruction, node clustering, multi-class node classification. Moreover, to verify the generalization ability of the proposed method, we consider four variants of ETINE: ETINE-D2V, ETINE-CNN, ETINE-BiRNN and ETINE-Bert, which employ the doc2vec, ATT-CNN, BiRNN and Bert composition methods for text modeling, respectively. The results of the three variants are concluded in the following experiments.

For a fair comparison, the default setting of dimension size is 200 for all models. According to [20,34], for DeepWalk, Node2vec, we set walk length as 40, window size as 10, and the number of walks per node as 80; besides, the return parameter p and in–out parameter q are set as $p = 0.25$, $q = 0.25$ for Node2vec. As mentioned in [58], for LINE, we set the mini-batch size of the stochastic gradient descent as 1, the learning rate of the starting value as 0.025, the number of negative samples as 5; and there we concatenate 100 dimensional first-order with 100 dimensional second-order representations to obtain 200 dimensional representation. For GraRep, we set the maximal transition step $K = 10$ and $\lambda = 1/N$. As suggested in [22], for TADW, we set the weight of regularization term as 0.2. In CANE, we set α, β, γ as 1, 0.3, 0.3, respectively, and the number of negative samples as 1. As suggested in [9], for COANE, the number of walks is set to 20, the walk length is set to 40 and the window size is 10. The

hyperparameters α and β of topic model are set to $50/k+1$ and $200/\omega$, respectively, where k is the number of communities and ω is the length of vocabulary. Margins appear moment M_m and margins enlarge speed M_s are set to 0.25 and 0.3, respectively. For our proposed model ETINE, we set the default parameters as follows: $\beta = 1/N$ for all datasets, maximal transition step $K = 5$ for Cora and CiteSeer-M6 datasets, and $K = 10$ for DBLP and Zhihu datasets. Moreover, according to the length of text, we set the size of vector $\overrightarrow{h}$ as 500 for Cora and Zhihu datasets, and 300 for CiteSeer-M6 and DBLP datasets. All the experiments are conducted at the platform of Intel(R) Xeon(R) CPU e5-2620 V3 @ 2.40 GHz processor with 24 cores and 128G memory, equipping a Tesla P40 with 24 GB vmemory. We use Tensorflow to implement the neural networks and OpenNE for manipulating the graph.

4.5. Multi-class node classification

We first investigate the performance of our ETINE in multi-class node classification. Specifically, we employ *logistic regression* to train classifiers on the labeled nodes that are randomly sampled from 10% to 90%. Then we test the remaining data to compute both *Micro-F1* and *Macro-F1* values, and report the statistical results over 10 independent trials. The classification results on Cora, CiteSeer-M6 and DBLP datasets are shown in Table 4, Table 5 and Table 6, respectively. The optimal results are highlighted in bold. From the results shown in these tables, we have the following observations and analyses:

- It is clearly shown that ETINEs significantly outperform nearly all baseline models on the tested benchmarks with different training ratios, which shows superior performance in node classifications. More specifically, ETINE-D2V obtains the best results on Cora dataset, and ETINE-CNN achieves the best performance on both CiteSeer-M6 and DBLP datasets. Although ETINE-BiRNN is slightly inferior than the other two variants, it still can achieve promising results in node classification. The above observations show that, ETINE-CNN employing ATT-CNN model for text modeling can help learn deeper semantics of nodes.

- Structure-and-Text models like TADW, CANE, ETINEs and COANE obviously outperform than structure-only baselines. Besides, compared to GraRep, ETINEs yield significant improvements on all the tested benchmarks. Moreover, ETINE-CNN yields at least 8.56% improvement on CiteSeer-M6 dataset when the training ratio reaches 90%. ETINE-D2V beats GraRep, TADW and COANE by 10.4%, 8.4% and 7.7%, respectively, on DBLP dataset when the training ratio reaches 90%. The experimental results demonstrate the importance of both network structure and text information, and the integration of structural and textual information is beneficial to learn more comprehensive network embeddings on big datasets.

- We also observe that the performance of TADW and CANE is not stable on the tested benchmarks. CANE performs well on Cora dataset but poorly on CiteSeer-M6 and DBLP datasets, because it involves much more parameters to train. TADW outperforms CANE on both the two benchmarks, since it can perform random walks to capture global structure of networks. On the other hand, we can observe that ETINEs are relatively stable on all of the baselines, which verifies its stability and robustness. In addition, CNN based methods, like ETINE-CNN and ETINE-Bert, are stable than the other variants of ETINEs.

Furthermore, in order to verify the performance of the RSVD approximation method, we further conduct a set of contrast experiments on Cora dataset to compare the results of ETINE using RSVD and SVD. The classification results are reported in Fig. 5, where the labeled nodes vary from 10% to 90%. We can observe that the classification results of ETINE with RSVD is superior to that of ETINE with SVD, which demonstrates that RSVD can improve the performance of our method with lower complexity.

4.6. Network reconstruction

An effective network embedding method should ensure that the learned embeddings can maintain most of the network information. Therefore, this section carries out the network reconstruction task to evaluate the performance of ETINE. We compute the values of *precision@h* with different h on all models. In practice, the links of the original network serve as the ground-truth, we want to predict the links of the network based on the learned node representations. The experimental results on DBLP and Zhihu datasets are shown in Fig. 6. Note that because there is no label information in the Zhihu dataset, it is only used for this task.

From Fig. 6 we can clearly find that, compared with other methods, the *precision@h* of ETINEs achieve substantial gains over other baselines. Specifically, on DBLP dataset, the *precision@h* of ETINE-D2V and ETINE-CNN can reach nearly 100% and maintain 100% from $h = 0$ to $h = 15\,000$; meanwhile, on Zhihu dataset, the *precision@h* of all the ETINE variants can keep 100% until h reaches to 10 000. It demonstrates that our method can almost perfectly reconstruct initial network on these two benchmark datasets. Moreover, compared with DeepWalk, LINE, Node2vec and GraRep that only exploit structure pattern, TADW and our ETINEs show superior performance. The premier reason may be that TADW incorporates text information into final embeddings. Despite this, ETINEs are also competitive with structure-and-text models TADW and CANE, which further demonstrates the effectiveness of our model.

4.7. Node clustering

In this section, we conduct node clustering task to verify the auxiliary role of textual information for learning network embeddings. For the task of node clustering, we take the node representations as input features, and feed them into *K-means* algorithm to conduct node clustering. Fig. 7 summarizes the average *NMI* values over 10 independent trials on Cora and CiteSeer-M6 datasets.

As is shown in Fig. 7, ETINEs, especially ETINE-CNN outperforms all the structure-only and structure-and-text baselines. In detail, ETINE-CNN outperforms CANE by 40.18% (*NMI*) and TADW by 28.14% on CiteSeer-M6 dataset. On the other hand, compared with structure-only methods, TADW and CANE achieve better performance on Cora dataset, while presenting poor results on CiteSeer-M6 dataset. GraRep performs well for node clustering on CiteSeer-M6 dataset, which indicates the structural proximity matrices can effectively capture global structure information. Compared with COANE which reaches the top among the baselines, ETINEs can also outperform COANE by 13.16% on Cora, and by 22.45%. Comparatively speaking, our proposed ETINE can effectively capture and model both structure-based and text-based information to learn high-quality network embeddings.

4.8. Parameter sensitivity

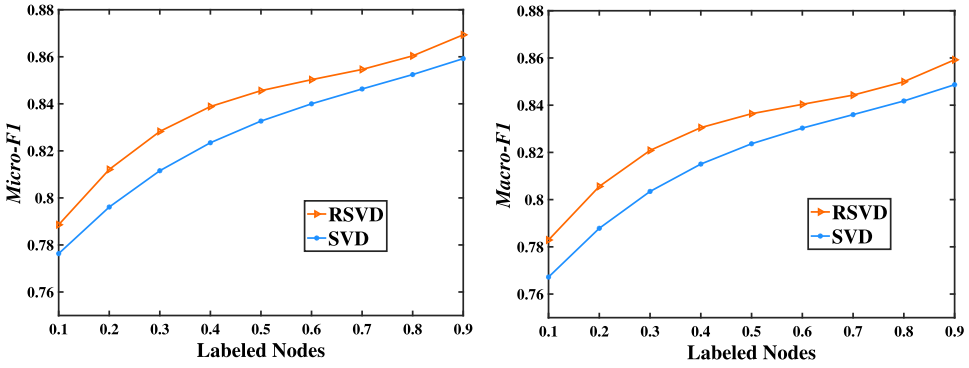
ETINE involves two critical hyperparameters: maximal transition step K and dimension size d. We run our model with

Table 5
Micro-F1 and Macro-F1 on CiteSeer-M6 dataset.

Metric	Model	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1	DeepWalk	77.88	80.32	81.71	82.73	83.00	83.53	83.59	83.63	83.66
	LINE	78.75	80.01	80.95	81.88	82.30	82.47	83.03	83.06	83.59
	GraRep	80.14	80.67	81.27	81.26	81.54	81.64	81.52	81.25	81.95
	Node2vec	79.11	80.70	81.64	82.28	82.72	83.16	83.55	83.80	83.59
	TADW	78.95	81.06	82.49	82.73	83.12	83.82	84.28	84.58	84.98
	CANE	72.81	75.13	75.89	76.53	77.01	77.13	77.53	77.86	77.09
	COANE	77.62	80.42	82.84	83.17	83.27	84.05	86.13	86.73	87.48
	ETINE-D2V	81.34	83.25	85.21	85.71	86.57	86.74	87.26	87.58	88.52
	ETINE-CNN	**86.40**	**87.41**	**87.96**	**88.46**	**88.81**	**89.16**	**89.57**	**89.74**	**90.30**
	ETINE-BiRNN	79.30	81.81	81.79	82.75	82.93	83.99	84.50	84.75	85.05
	ETINE-Bert	85.20	87.05	88.41	88.53	88.94	89.28	89.39	89.69	89.70
Macro-F1	DeepWalk	77.42	79.93	81.40	82.45	82.69	83.25	83.31	83.33	83.27
	LINE	78.39	79.66	80.59	81.55	81.98	82.17	82.71	82.75	83.24
	GraRep	79.76	80.24	80.93	80.97	81.20	81.31	81.28	81.04	81.64
	Node2vec	78.89	80.45	81.44	82.08	82.52	82.97	83.37	83.62	83.31
	TADW	79.11	81.22	82.64	82.87	83.28	84.00	84.46	84.72	85.03
	CANE	72.25	74.65	75.47	76.53	76.61	76.65	76.94	77.31	76.53
	COANE	77.35	80.38	83.3	83.37	83.64	84.25	85.95	87.15	88.09
	ETINE-D2V	81.22	83.56	85.05	85.57	86.41	86.56	87.07	87.38	88.26
	ETINE-CNN	**86.39**	**87.42**	**87.97**	**88.44**	**88.80**	**89.15**	**89.52**	**89.67**	**90.20**
	ETINE-BiRNN	80.22	81.73	82.61	83.16	83.74	83.92	84.41	84.84	85.68
	ETINE-Bert	85.22	87.13	88.12	88.63	89.02	89.36	89.50	89.78	89.69

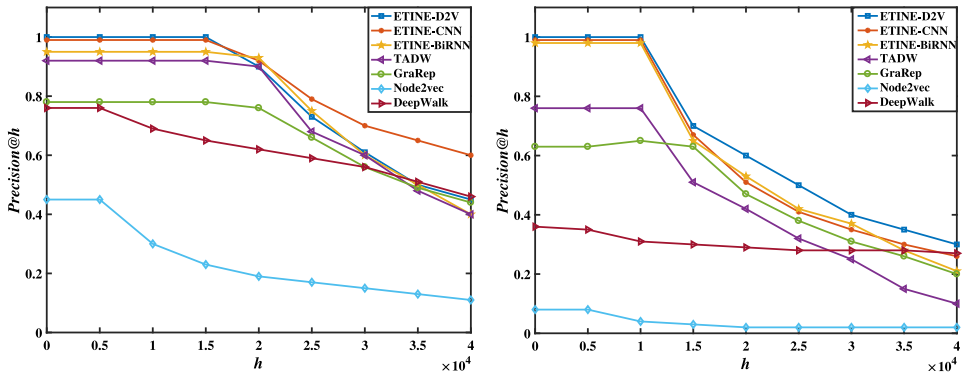
Table 6
Micro-F1 and Macro-F1 on DBLP dataset.

Metric	Model	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1	DeepWalk	73.64	74.83	75.25	75.56	75.67	75.65	75.85	75.81	75.66
	LINE	80.79	81.05	81.15	81.08	81.17	81.27	81.32	81.23	81.04
	GraRep	80.49	80.96	80.96	80.97	81.00	81.03	81.07	81.33	81.30
	Node2vec	79.23	80.08	80.31	80.48	80.60	80.58	80.49	80.61	80.53
	TADW	81.01	81.75	82.07	82.20	82.47	82.63	82.79	82.83	82.77
	CANE	79.98	81.08	81.51	81.87	82.23	82.34	82.42	82.53	82.53
	COANE	81.30	82.14	82.62	83.10	83.66	83.84	84.20	84.42	84.38
	ETINE-D2V	**85.78**	**87.28**	**88.02**	**88.36**	**88.58**	**88.74**	**88.82**	**89.10**	**89.17**
	ETINE-CNN	83.40	83.90	84.08	84.22	84.25	84.32	84.43	84.41	84.53
	ETINE-BiRNN	78.92	79.65	80.63	80.41	82.60	82.89	82.95	83.50	83.25
	ETINE-Bert	81.89	82.54	82.72	82.91	83.07	83.14	83.18	83.96	83.86
Macro-F1	DeepWalk	65.96	67.15	67.51	67.87	68.05	68.02	68.34	68.29	68.12
	LINE	75.67	75.99	76.22	76.13	76.25	76.38	76.45	76.18	76.06
	GraRep	74.85	75.42	75.40	75.41	75.42	75.36	75.39	75.68	75.61
	Node2vec	73.13	74.08	74.50	74.86	74.99	75.00	74.94	75.13	74.81
	TADW	75.11	76.17	76.73	76.89	77.29	77.49	77.81	77.72	77.60
	CANE	72.74	74.37	75.09	75.58	76.07	76.17	76.38	76.53	76.20
	COANE	75.95	77.15	77.74	78.50	79.15	79.33	79.81	80.23	79.99
	ETINE-D2V	**81.26**	**83.27**	**84.01**	**84.58**	**84.88**	**85.09**	**85.24**	**85.59**	**85.95**
	ETINE-CNN	78.52	79.39	79.62	79.76	79.76	79.76	79.90	79.88	80.06
	ETINE-BiRNN	75.42	76.55	77.13	77.43	77.93	78.03	78.55	78.86	78.62
	ETINE-Bert	76.84	77.60	77.91	78.20	78.41	78.60	78.66	78.34	78.29



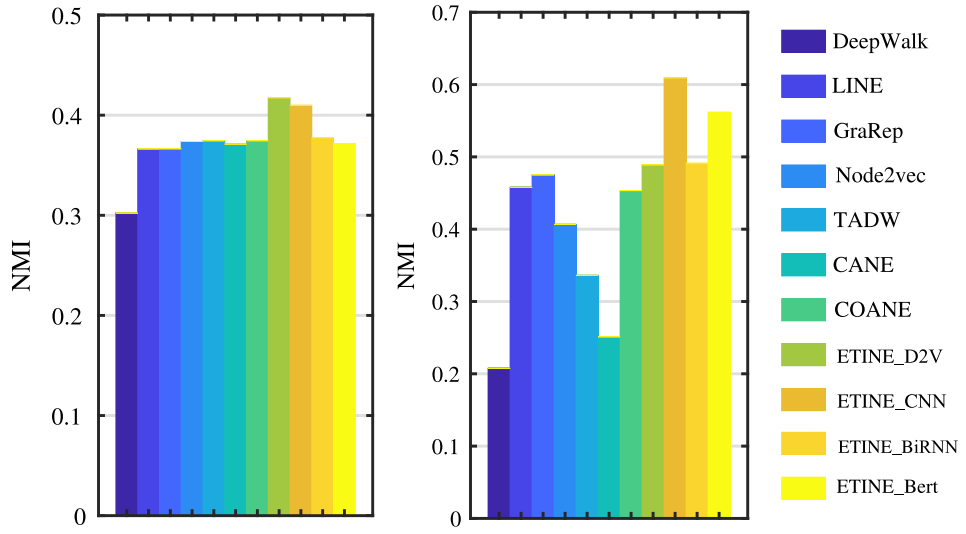
(a) *Micro-F1* (b) *Macro-F1*

Fig. 5. Classification results of different approximation methods RSVD and SVD on Cora dataset with Doc2vec.



(a) DBLP dataset (b) Zhihu dataset

Fig. 6. *Precision@h* between different models on (a) DBLP dataset and (b) Zhihu dataset.



(a) Cora dataset (b) CiteSeer-M6 dataset

Fig. 7. Average *NMI* values between different models on (a) Cora dataset and (b) CiteSeer-M6 dataset.

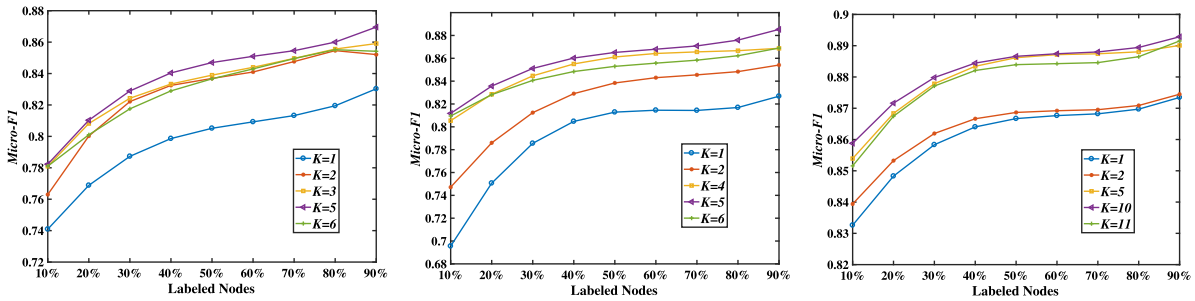
different K and d to assess their impact on the experimental results.

Figs. 8 and 9 show the sensitivity of different K on Cora, CiteSeer-M6 and DBLP datasets. *Micro-F1* and *Macro-F1* are taken as evaluation metrics when the training ratio changes from 10% to 90%. Here we set d to 200. As shown in Figs. 8(a) and 9(a), on the Cora dataset, the results of setting $K = 2$ has an encouraging improvement over $K = 1$, and the results of $K = 3$ further exceeds $K = 2$. When $K = 5$, ETINE obtains the best performance. However, the performance of $K = 6$ is no greater than $K = 5$. It verifies that when K is big enough, the quality of the learned node representations tends to be steady, which is mentioned in Section 3.3.1. For legibility, the results of $K = 4$ is not shown because the results of $K = 4$ is comparable to $K = 3$. Similarly, Figs. 8(b) and 9(b), as well as Figs. 8(c) and 9(c) show the performances of different K on CiteSeer-M6 and DBLP datasets, respectively. Specifically, on CiteSeer-M6 dataset, the results of $K = 5$ achieves optimal performance. Moreover, the results of $K = 4$ slightly outperforms $K = 3$, thus the results of $K = 4$ is not given for readability. On DBLP dataset, obviously, when $K = 10$, ETINE yields the optimal results. In the same way, compared with the results of $K = 2$, the results of $K = 3, 4$ have slight improvement. Meanwhile, the results of $K = 6, 7, 8, 9$ are competitive with $K = 5$. Therefore, the results of $K = 3, 4, 6, 7, 8, 9$ are not shown.

Fig. 10 shows *Micro-F1* and *Macro-F1* values when d increases from 50 to 300. In the experiments, we set training ratio to 10%, and we set the default setting as follows: $K = 5$ for Cora and CiteSeer-M6 datasets, and $K = 10$ for DBLP dataset. Interestingly, for all datasets, there is a slightly increase in results when d changes from 100 to 200. And the results reach the top when d reaches 200. However, if we continue to increase d to larger values, the classification performance remains stable on the whole.

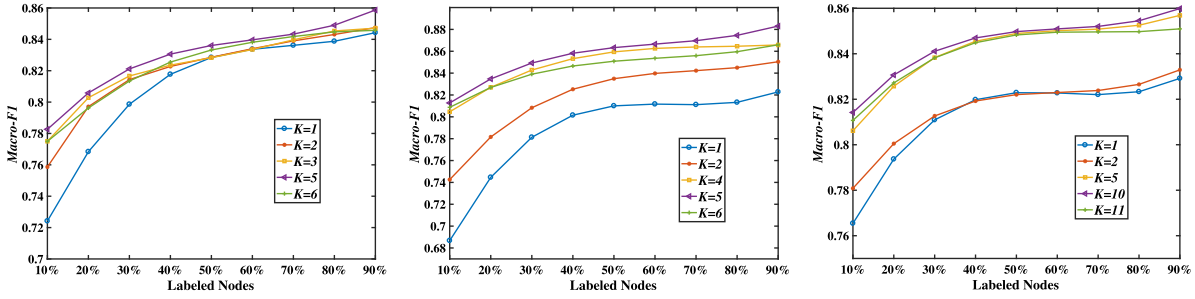
5. Conclusion and future work

In this paper, we present an Enhanced Textual Network Embedding (ETINE) method to integrate network structure and text information by independently and comprehensively modeling structural and textual proximities of the information network. Specifically, the proposed textual proximity matrix uses probabilistic sampling method to preserve deep semantic information of the text attributes. We define the optimization of our proposed structure-based and text-based loss functions as a matrix approximation problem. Additionally, instead of using traditional low-rank approximation, we apply a more efficient matrix approximation method (RSVD) to optimize the structure-based and text-based objectives. Empirically, we evaluate the performance



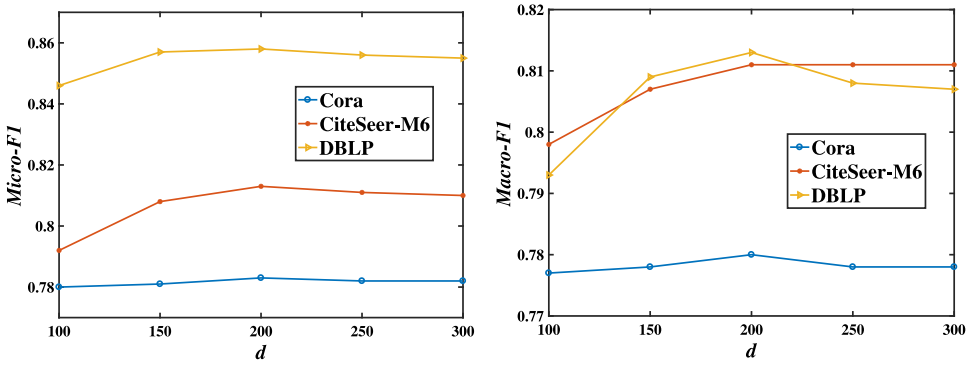
(a) *Micro-F1* on Cora dataset (b) *Micro-F1* on CiteSeer-M6 dataset (c) *Micro-F1* on DBLP dataset

Fig. 8. *Micro-F1* with different K.



(a) *Macro-F1* on Cora dataset (b) *Macro-F1* on CiteSeer-M6 dataset (c) *Macro-F1* on DBLP dataset

Fig. 9. *Macro-F1* with different K.



(a) *Micro-F1* (b) *Macro-F1*

Fig. 10. Results with different d.

of ETINE on four real-world textual information networks. Experimental results demonstrate that ETINE successfully encodes the both structural and textual information into node representations and shows superior performance compared with our baselines.

In the future, a promising direction is to explore the online learning models of ETINE in large-scale networks. Besides, we can extend our ETINE model to handle networks with different types of information, e.g. images and labels.

CRediT authorship contribution statement

Wenfeng Liu: Conceptualized the problem and the technical framework, Developed the algorithms and conducted the experiments and exported the data, Writing - original draft, Discussed the experimental results and commented on the manuscript. **Maoguo Gong:** Conceptualized the problem and the technical framework, Writing - original draft, Discussed the experimental results and commented on the manuscript. **Zedong Tang:** Developed the algorithms and conducted the experiments and exported the data, Writing - original draft, Discussed the experimental results and commented on the manuscript.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant no. 61772393), the National key research and development program of China (Grant no. 2017YFB0802200), and the Key research and development program of Shaanxi Province (Grant no. 2018ZDXM-GY-045).

References

[1] L. Liao, X. He, H. Zhang, T. Chua, Attributed social network embedding, IEEE Trans. Knowl. Data Eng. 30 (12) (2018) 2257–2270.

[2] Z. Zhao, X. Zhang, C. Li, F. Chiclana, E. Herreraviedma, An incremental method to detect communities in dynamic evolving social networks, Knowl.-Based Syst. 163 (2019) 404–415.

[3] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, Z. Su, ArnetMiner: extraction and mining of academic social networks, in: Proceedings of the 15rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, pp. 990–998.

[4] X. Lei, M. Fang, H. Fujita, Moth–flame optimization-based algorithm with synthetic dynamic PPI networks for discovering protein complexes, Knowl.-Based Syst. 172 (2019) 76–85.

[5] L.F. Ribeiro, P.H. Saverese, D.R. Figueiredo, struc2vec: Learning node representations from structural identity, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 385–394.

[6] D. Zhang, J. Yin, X. Zhu, C. Zhang, Network representation learning: A survey, IEEE Trans. Big Data 6 (1) (2020) 3–28.

[7] D. Zhu, P. Cui, Z. Zhang, J. Pei, W. Zhu, High-order proximity preserved embedding for dynamic networks, IEEE Trans. Knowl. Data Eng. 30 (11) (2018) 2134–2144.

[8] Y. Xie, C. Yao, M. Gong, C. Chen, A.K. Qin, Graph convolutional networks with multi-level coarsening for graph classification, Knowl.-Based Syst. 194 (2020) 105578.

[9] Y. Gao, M. Gong, Y. Xie, H. Zhong, Community-oriented attributed network embedding, Knowl.-Based Syst. 193 (2020) 105418.

[10] N. Guan, D. Song, L. Liao, Knowledge graph embedding with concepts, Knowl.-Based Syst. 164 (2019) 38–44.

[11] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1225–1234.

[12] M. Ou, P. Cui, J. Pei, Z. Zhang, W. Zhu, Asymmetric transitivity preserving graph embedding, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1105–1114.

[13] F.D. Malliaros, M. Vazirgiannis, Clustering and community detection in directed networks: A survey, Phys. Rep. 533 (4) (2013) 95–142.

[14] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, AI Mag. 29 (3) (2008) 93–106.

[15] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Proceedings of the 5th International Conference on Learning Representations, 2016.

[16] W.L. Hamilton, R. Ying, J. Leskovec, Representation learning on graphs: Methods and applications, IEEE Data(base) Eng. Bull. 40 (2017) 52–74.

[17] S. Cao, W. Lu, Q. Xu, Deep neural networks for learning graph representations, in: Proceedings of the 30th AAAI Conference on Artificial Intelligence, 2016, pp. 1145–1152.

[18] S. Cavallari, V.W. Zheng, H. Cai, K.C.-C. Chang, E. Cambria, Learning community embedding with community detection and node embedding on graphs, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, ACM, 2017, pp. 377–386.

[19] C. Tu, H. Wang, X. Zeng, Z. Liu, M. Sun, Community-enhanced network representation learning for network analysis, 2016, arXiv preprint arXiv: 1611.06645.

[20] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 855–864.

[21] P. Goyal, E. Ferrara, Graph embedding techniques, applications, and performance: A survey, Knowl.-Based Syst. 151 (2018) 78–94.

[22] C. Yang, Z. Liu, D. Zhao, M. Sun, E.Y. Chang, Network representation learning with rich text information, in: Proceedings of the 24th International Joint Conference on Artificial Intelligence, 2015, pp. 2111–2117.

[23] X. Sun, J. Guo, X. Ding, T. Liu, A general framework for content-enhanced network representation learning, 2016, arXiv preprint arXiv:1610.02906.

[24] C. Tu, H. Liu, Z. Liu, M. Sun, Cane: Context-aware network embedding for relation modeling, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vol. 1, 2017, pp. 1722–1731.

[25] S. Cao, W. Lu, Q. Xu, Grarep: Learning graph representations with global structural information, in: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, 2015, pp. 891–900.

[26] C. Tu, W. Zhang, Z. Liu, M. Sun, et al., Max-margin DeepWalk: Discriminative learning of network representation, in: Proceedings of the 25th International Joint Conference on Artificial Intelligence, 2016, pp. 3889–3895.

[27] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, S. Yang, Community preserving network embedding, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017, pp. 203–209.

[28] M. Fazel, Matrix Rank Minimization with Applications (Ph.D. thesis), Stanford University, 2002.

[29] H. Abdi, L.J. Williams, Principal component analysis, Wiley Interdiscip. Rev. Comput. Stat. 2 (4) (2010) 433–459.

[30] D.D. Lee, H.S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (6755) (1999) 788–791.

[31] N. Halko, P.-G. Martinsson, J.A. Tropp, Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, SIAM Rev. 53 (2) (2011) 217–288.

[32] W. He, H. Zhang, L. Zhang, H. Shen, Hyperspectral image denoising via noise-adjusted iterative low-rank matrix approximation, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 8 (6) (2015) 3050–3061.

[33] S. Wang, C. Zhang, H. Qian, Z. Zhang, Using the matrix ridge approximation to speedup determinantal point processes sampling algorithms, in: Proceedings of the 28th AAAI Conference on Artificial Intelligence, 2014, pp. 2121–2127.

[34] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 701–710.

[35] Y. Dong, N.V. Chawla, A. Swami, metapath2vec: Scalable representation learning for heterogeneous networks, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 135–144.

[36] L. Gui, Y. Zhou, R. Xu, Y. He, Q. Lu, Learning representations from heterogeneous network for sentiment classification of product reviews, Knowl.-Based Syst. 124 (2017) 34–45.

[37] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations ofwords and phrases and their compositionality, in: Proceedings of 27th Annual Conference on Neural Information Processing Systems, Vol. 26, 2013, pp. 3111–3119.

[38] X. Sun, Z. Song, J. Dong, Y. Yu, C. Plant, C. Bohm, Network structure and transfer behaviors embedding via deep prediction model, in: Proceedings of the 33th AAAI Conference on Artificial Intelligence, Vol. 33, No. 01, 2019, pp. 5041–5048.

[39] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xing, M. Guo, GraphGAN: Graph representation learning with generative adversarial nets, in: Proceedings of the 32th AAAI Conference on Artificial Intelligence, 2018, pp. 2508–2515.

[40] Y.-C. Lee, N. Seo, K. Han, S.-W. Kim, ASiNE: Adversarial signed network embedding, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, USA, 2020, pp. 609–618.

[41] B. Yu, B. Lu, C. Zhang, C. Li, K. Pan, Node proximity preserved dynamic network embedding via matrix perturbation, Knowl.-Based Syst. 196 (2020) 105822.

[42] S. Pan, J. Wu, X. Zhu, C. Zhang, Y. Wang, Tri-party deep network representation, in: Proceedings of the 25th International Joint Conference on Artificial Intelligence, 2016, pp. 1895–1901.

[43] P. Cheng, Y. Li, X. Zhang, L. Chen, D. Carlson, L. Carin, Dynamic embedding on textual networks via a Gaussian process, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pp. 7562–7569.

[44] W. Huang, Y. Li, Y. Fang, J. Fan, H. Yang, BiANE: Bipartite attributed network embedding, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 149–158.

[45] G. Cui, J. Zhou, C. Yang, Z. Liu, Adaptive graph encoder for attributed graph embedding, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2020, pp. 976–985.

[46] Y. Chen, J. Zheng, D. Li, Sparse attributed network embedding via adaptively aggregating neighborhood information, in: Proceedings of the IEEE International Joint Conference on Neural Networks, IJCNN, 2020, pp. 1–8.

[47] Z. Ali, G. Qi, K. Muhammad, B. Ali, W.A. Abro, Paper recommendation based on heterogeneous network embedding, Knowl.-Based Syst. 210 (2020) 106438.

[48] C. Zhang, G. Wang, B. Yu, Y. Xie, K. Pan, Proximity-aware heterogeneous information network embedding, Knowl.-Based Syst. 193 (2020) 105468.

[49] M.U. Gutmann, A. Hyvarinen, Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics, J. Mach. Learn. Res. 13 (1) (2012) 307–361.

[50] P.D. Turney, Domain and function: A dual-space model of semantic relations and compositions, J. Artificial Intelligence Res. 44 (2012) 533–585.

[51] Q. Le, T. Mikolov, Distributed representations of sentences and documents, in: Proceedings of the 31th International Conference on Machine Learning, 2014, pp. 1188–1196.

[52] N. Kalchbrenner, E. Grefenstette, P. Blunsom, A convolutional neural network for modelling sentences, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 2014, pp. 655–665.

[53] M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (11) (1997) 2673–2681.

[54] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[55] O. Levy, Y. Goldberg, Neural word embedding as implicit matrix factorization, in: Proceedings of 28th Annual Conference on Neural Information Processing Systems, 2014, pp. 2177–2185.
[56] A. Mccallum, K. Nigam, J.D.M. Rennie, K. Seymore, Automating the construction of internet portals with machine learning, Inf. Retr. 3 (2) (2000) 127–163.
[57] W. Liu, M. Gong, S. Wang, L. Ma, A two-level learning strategy based memetic algorithm for enhancing community robustness of networks, Inform. Sci. 422 (2018) 290–304.
[58] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in: Proceedings of the 24th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.