mathematics

Article

An Optimized Network Representation Learning Algorithm Using Multi-Relational Data

Zhonglin Ye [1,2,3,4], Haixing Zhao [1,2,3,4,*], Ke Zhang [1,2,3], Yu Zhu [1,2,3] and Zhaoyang Wang [1,2,3]

1 School of Computer, Qinghai Normal University, Xining 810800, China; zhonglin_ye@foxmail.com(Z.Y.); hbsanli@163.com (K.Z.); sdcxzhuyu@163.com (Y.Z.); wzy02130822@163.com (Z.W.)
2 Tibetan Information Processing and Machine Translation Key Laboratory of Qinghai Province, Xining 810008, China
3 Key Laboratory of Tibetan Information Processing, Ministry of Education, Xining 810008, China
4 School of Computer Science, Shaanxi Normal University, Xi'an 710119, China
* Correspondence: h.x.zhao@163.com

Received: 26 April 2019; Accepted: 16 May 2019; Published: 21 May 2019

Abstract: Representation learning aims to encode the relationships of research objects into low-dimensional, compressible, and distributed representation vectors. The purpose of network representation learning is to learn the structural relationships between network vertices. Knowledge representation learning is oriented to model the entities and relationships in knowledge bases. In this paper, we first introduce the idea of knowledge representation learning into network representation learning, namely, we propose a new approach to model the vertex triplet relationships based on DeepWalk without TransE. Consequently, we propose an optimized network representation learning algorithm using multi-relational data, MRNR, which introduces the multi-relational data between vertices into the procedures of network representation learning. Importantly, we adopted a kind of higher order transformation strategy to optimize the learnt network representation vectors. The purpose of MRNR is that multi-relational data (triplets) can effectively guide and constrain the procedures of network representation learning. The experimental results demonstrate that the proposed MRNR can learn the discriminative network representations, which show better performance on network classification, visualization, and case study tasks compared to the proposed baseline algorithms in this paper.

Keywords: network representation; network embedding; representation learning; knowledge representation; joint learning

1. Introduction

Network representation learning (NRL) plays a pivotal role in many areas, which aims at learning low-dimensional, compressed, and distributed representation vectors for all kinds of networks. Network representation learning tasks can be intuitively regarded as network encoding tasks, where each node is given a unique vector in representation space, consequently, the neighboring vertices possess a closer space distance based on the distance evaluation function. Distributed representation learning is derived from a language-embedding application. The representative algorithm of language embedding is Word2Vec, proposed by Mikolov et al. [1,2]. Motivated by the Word2Vec algorithm, the network representation learning algorithm was consequently proposed by Perozzi [3], who adopted the random walk strategy to generate the vertex sequences, which are the same as "sentences" in language models. DeepWalk receives multiple vertex sequences $v_0, v_1 \ldots, v_n$ as the input of the network representation learning model, where n denotes the length of the random walk sequence. In the middle layer of the learning model, the sequence $v_0, v_1 \ldots, v_n$ is converted into the vertex pairs (v_0, v_{-2}),

(v_0, v_{-1}), (v_0, v_1), and (v_0, v_2), where the window size needs to be set and the number of vertex pairs is controlled by the window size. The output of DeepWalk is a low-dimensional vector, $r_v \in \mathbb{R}^k$, where k is the size of the network representation vector. Recently, the DeepWalk algorithm has been successfully applied to many tasks [4].

There exist various relationships between network vertices. The structural relationships can be captured by DeepWalk and the community relationships can be captured by community detection algorithms or vertex labels. The vertex relationships in real-world networks also can be intuitively captured by the rich vertex contents (texts). For example, different titles of network vertices may contain the same words in citation networks, so the higher the overlap of words, the greater the probability of reference between them. Indeed, it is also well acknowledged that most existing NRL models ignore the semantic relationships between vertices. However, some NRL models try to integrate node contents into network representation learning. For example, text-associated deep walk (TADW) [5] and tri-party deep network representation (TriDNR) [6] provide a novel NRL framework, which integrates text features into representations by matrix factorization and joint learning. More specifically, TADW adopts inductive matrix completion (IMC) [7] to integrate text features into a network representation model; the essence of TADW is a kind of matrix factorization. Thus, it is not suitable for tasks of large-scale network representation learning. Two neural networks are applied by TriDNR to integrate text contents and label information from network vertices into a network representation learning model, which learns network representations in three parts (i.e., node structure, node content, and node label), consequently, TriDNR captures the inter–node, node–word, and label–word relationships. Because it considers the relationships between inter–node, node–word, and label–word, TriDNR has a higher computational complexity, and it would take a longer training time for representation learning, even in some smaller networks. Therefore, the objective of this paper is to research how to make full use of the network node text contents and how to improve the performance of network representation learning by node text contents. Finally, we hope that the proposed algorithm can obtain smaller computational complexity in tasks of large-scale network representation learning and can also gain a higher precision and desirable performance.

Multi-relational data modeling was first proposed by Bordes et al. [8], and it is a directed graph containing entities and relationships such as triplet form (head, label, tail). The triplet is denoted in the form (h, l, t) in this paper. An example is knowledge bases, such as Freebase [9] and DBpedia [10], where each entity is an abstract concept and the relationships are the facts involving two entities. TransE [8] can model multi-relational data using more generic methods that take all heterogeneous relationships into account at the same time. The TransE algorithm mainly learns knowledge embedding based on Freebase and DBpedia, etc.

The feature matrix of the network is factorized by TADW to gain the node representations, where the text feature matrix is as an auxiliary matrix. Note that the node contents are converted to a text feature matrix in TADW. This kind of TriDNR algorithm considers words in node contents as kind of special vertices and models node–word and word–label relationships, where all words in node contents are input into the neural network to train and learn the network representation vectors. Thus, we want to propose a novel network representation learning algorithm, which can model inter–vertex and vertex–word relationships with a lower calculation complexity. More importantly, the node representations can possess semantic information. To achieve the above goal, we first constructed the vertex relationships by node contents in this paper. We then converted these vertex relationships into vertex triplets in the form (h, l, t), where l denotes the semantic relationship between vertex h and vertex t. Consequently, we propose an optimized network representation learning algorithm using vertex triplets, called MRNR, which focuses on modeling the network structure and multi-relational data at the same time. Motivated by the TransE model, we introduce the idea of the TransE algorithm to optimize the network representation learning process and adopt DeepWalk to model network structure features. Please note that we propose a new approach based on DeepWalk to model all knowledge triplets. Through this we learn the representations of triplets (h, l, t) using the improved algorithm without TransE. The TransE [8] and DeepWalk [3] are all proposed based on the Word2Vec [1,2]. So, we propose a unified joint learning framework and objective function without concatenating two kinds of vertex representations. In this framework, MRNR learns the network representations and model the multi-relational data using a single neural network (TriDNR uses two neural networks). Moreover, only some words in the node text contents are involved in the tasks of network representation learning. Therefore, MRNR is competent to the tasks of the large-scale network representation learning. Finally, we introduce a kind of optimization algorithm to improve the performance of network representation learning. However, MRNR is a novel node semantic relationship preserving network representation algorithm. By jointly optimizing the DeepWalk structure preserving objective and TransE [8] node semantic relation preserving objective, the proposed MRNR algorithm learns informative node representations.

To summarize, we make the following contributions:

(1) We propose a novel vertex triplet-based NRL model, MRNR. It incorporates the multi-relational data between vertices into the network representation learning model. As a simple and straightforward model, it models the network representations between vertices and vertex triplets with a unified joint learning framework at the same time.
(2) To the best of our knowledge, we are the first to introduce the idea of a vertex triplet into network representation learning, which means that the external multi-relational data are introduced into DeepWalk as a regularization term to guide and constrain the training procedure. Moreover, we introduce a kind of higher order transformation strategy to optimize the learnt network representation vectors.
(3) We conduct experiments on the three real-world datasets with the task of node classification, the experimental results show that the MRNR is an effective and robust network representation approach. Importantly, MRNR can achieve excellent performance in large-scale network learning tasks, especially in terms of accuracy and time complexity.

2. Related Work

Representation learning [11] algorithms aim at avoiding huge feature engineering and improving the application areas of network features. Representation learning algorithms learn features via neural network or matrix factorization methods. DeepWalk [3] is a network representation learning algorithm based on Word2Vec [1,2]. DeepWalk [3] can encode complex network structures into the low-dimensional representation vector space, showing better network classification performance on various network tasks.

Consequently, some improved algorithms are proposed for network representation learning tasks. Several are based on DeepWalk [3], such as node2vec [12], NEU [13], WALKLETS [14] and HARP [15] etc., while node2vec [12] optimizes the strategy of random walk. For example, node2vec [12] adopts breadth first search and depth first search to sample the node sequences. The breadth first search provides a local microscopic view and depth first search provides a global macroscopic view. DeepWalk [3] can capture node relationships of lower order, and NEU [13] introduces a novel approach, which can obtain higher order network representations using simple representation transformation. Moreover, the NEU [13] algorithm implicitly approximates higher order proximities with theoretical approximation bound, WALKLETS [14] obtains higher order representations by the *k*-step random walk strategy, and HARP [15] learns higher order representations by graph compression strategy. The low order features are just to consider the local relations amongst vertices, and the high order features mainly take the global relations amongst vertices into account.

Another kind of improvement is based on neural matrix factorization of DeepWalk [3], such as Line [16], GraRep [17], TADW [5], MMDW [18]. Specifically, the Line [16] fully researches the 1st order proximity and the 2nd order proximity based on matrix factorization. DeepWalk [1] actually aims at factorizing the structure feature matrix of the networks. Consequently, GraRep [17] introduces a matrix factorization strategy to obtain higher order representations of the networks. GraRep [17] obtains higher order features of networks through multiplication of adjacency matrix. The uses inductive matrix completion (IMC) [7] is adopted by TADW [5] to integrate the node contexts into network representations. The max-margin method is used by MMDW [18] to optimize the learnt network representations. In fact, both TADW [5] and MMDW [18] adopt the matrix factorization approach to obtain the network representations.

The third kind of improvement is based on the joint learning model, such as DDRW [19], TriDNR [6], GCN [20], M-NMF [21], AANE [22], COSINE [23], and comE, [24] etc. Specifically, the DDRW is an improved algorithm based on DeepWalk [3], and DDRW [19] jointly improves the objective of classification and the objective of embedding entities. Two neural networks are adopted by TriDNR to learn the network representations from three parts (the node's structure, node's content, and node's label), consequently, TriDNR [6] captures the inter–node, node–word, and label–word relationships. Compared with TADW [5], the relationships of the node–label could be learnt based on these two neural networks. Both the local graph structures and features of the nodes can be encoded by GCN [20] by learning the hidden layer representations. And M-NMF [21] is a modularized non-negative matrix factorization approach, which incorporates the community structures into network representations. Both COSINE [23] and comE [24] also incorporate the community information into network representations. In addition, AANE [22] is an accelerated attributed network embedding approach, which enables the joint learning framework to be conducted by decomposing the complicated learning and training into many sub-problems.

There exist some network representation approaches for special network structures, such as DynamicTriad [25], DepthLGP [26], and DHNE [27], etc. In addition, motivated by generative adversarial networks (GANs) [28], some network representation learning algorithms introduce GANs to optimize the learning procedure, such as ANE [29], GraphGAN [30], and NetGan [31].

To sum up, various solutions and novel algorithms are proposed for tasks of network representation learning. The focus of this paper is to research how to efficiently use network node contents to improve the performance of network representation learning. More importantly, we research how to use the network node contents to conduct learning and training tasks of network representation on large-scale networks, which can overcome higher calculation complexity like TADW [5] and TriDNR [6] algorithms. We mainly use TADW [5] and MRNR [6] proposed in this paper to make a comparative analysis in a fair way, as these algorithms only adopt node structures and node contents to learn network representation vectors.

3. Method

The MRMR algorithm proposed in this paper aims to embed the text features of network vertices into the network representation vectors. Some network representation learning methods based on matrix factorization take the text features of network vertices as the feature matrix to optimize the factorization procedures of the network structure feature matrix, for example, the TADW algorithm belongs to this kind of NRL algorithm. Other network representation learning methods based on neural networks simultaneously model the relationships between vertices and the relationships between vertices and text through multiple neural networks, for example, the TriDNR algorithm belongs to this kind of NRL algorithm. Different from the TADW and TriDNR algorithms, a novel text-embedding method was adopted in this paper, namely, the text between network vertices was transformed into the triplets between vertices. In order to embed vertex triplets into the network representation learning task, we introduced the idea of multi-relationship modeling, namely, we adopted the idea of knowledge-embedding learning using TransE to model the triplet relationships between network vertices. In order to constrain the modeling of network vertex relationships by modeling vertex triplet relationships, we propose the objective function of the MRNR algorithm based on the CBOW model optimized by the negative sampling and TransE algorithms. Therefore, we first introduce how to construct the triplets between vertices in this section, then introduce the modeling procedure of the TransE algorithm, and then we introduce the CBOW model based on negative sampling optimization, and finally we introduce how the MRNR algorithm proposed in this paper can simultaneously model the network vertex relationships and triplet relationships.

In Section 3.1, we introduce some definitions of the parameters used in this paper. In Section 3.2, we introduce how to construct the vertex triplets based on text contents of two vertices. In Sections 3.3 and 3.4, the TransE algorithm and CBOW algorithm are introduced. After that, we propose the MRNR algorithm based on the improvements of these two algorithms in Section 3.5. Therefore, we introduce in detail how to embed vertex triplets into the network representation learning procedure based on the idea of the TransE algorithm in Section 3.5, and how to optimize the learnt network representation vectors obtained from the MRNR algorithm in Section 3.6.

3.1. Formalization

The parameters used in this paper are defined as follows. For a network $G = (V, E)$, we denote V as the vertex set, E as the edge set, and $|V|$ as the size of vertex set, where each vertex $v_0, v_1 \ldots, v_n \in V$ and each edge $e_0, e_1 \ldots, e_m \in E$. The purpose of the MRNR is to train the representation $r_v \in \mathbb{R}^k$ for a given vertex $v \in V$, which is also the goal of network representation learning. Here, k is the size of network representation vector and it is expected to be much smaller than $|V|$. Namely, $|V|$ usually denotes the number of vertices in the network, and we often set the size of the network representation vector as 100, 200, 300, etc., which is much smaller than the number of vertices. Note that $r_v \in \mathbb{R}^k$ is not task-specific and it can be used for various tasks. Because of the output of the DeepWalk algorithm, so can the learnt network representation vectors $r \in \mathbb{R}^k$ be used for network node classification, network visualization, link prediction, and recommendation systems, etc. For the multi-relational data modeling, we set the (h, l, t) as vertex triplets, where h denotes the head entity, t denotes the tail entity, and l denotes the relations between head entity and tail entity. For the CBOW model in DeepWalk, we set the $Context(v)$ as the context vertices of the current vertex v and $NEG(v)$ denotes the negative sampling procedure for the current vertex v. Here, the high-frequency variables are defined and explained, and the other variables adopted by this paper will be explained in detail at the corresponding sections.

3.2. Triplet Construction

The semantic relationships between vertices are extracted from the vertex texts. In this paper, we take two kinds of strategies to denote the relationships between vertices. The first kind of relationship is called as Unitary Relationship, where the relationship is constructed by one same word amongst word sequences from two vertex's texts. The second kind of relationship is called as Binary Relationship, where the relationship is constructed by two same words amongst the word sequences from two vertex's texts. We give a simple example in Table 1.

Table 1. Example of a unitary and binary relationship.

Node Text	**Node1: Online Learning of Multi-Scale Network Embedding** **Node2: Large-Scale Information Network Embedding**	**Total**
Unitary Relationship	(1, Network, 2), (1, Embedding, 2)	2
Binary Relationship	(1, Network_Embedding, 2)	1

3.3. TransE Model

Inspired by Word2Vec [1,2], Bordes et al. [8] proposed the TransE algorithm, and the TransE algorithm models the relationships by training the triplets (head, label, tail) (denoted by (h, l, t)) of multi-relational data as a form of translation tasks. It means that TransE needs to continuously adjust the representations of parameters h, l, and t to get a global equilibrium, $\boldsymbol{h} + \boldsymbol{l} \approx \boldsymbol{t}$. The objective function of TransE is as follows:

$$L = \sum_{(h,l,t)\in S} \sum_{(h',l,t')\in S'_{(h,l,t)}} [\gamma + d(\boldsymbol{h} + \boldsymbol{l}, \boldsymbol{t}) - d(\boldsymbol{h}' + \boldsymbol{l}, \boldsymbol{t}')]_+ \tag{1}$$

where

$$S'_{(h,l,t)} = \{(h', l, t)\big|h' \in E\} \cup \{(h, l, t')\big|t' \in E\}. \tag{2}$$

Here, S is the set of triplets (h, l, t) and S' is the set of negative samples. The energy of a triplet is equal to $d(\boldsymbol{h} + \boldsymbol{l}, \boldsymbol{t})$ for the dissimilarity measure and d, $\gamma > 0$ is a margin hyperparameter. $[x]_+$ is the positive part of variable x.

3.4. CBOW Using Negative Sampling

DeepWalk can be implemented by CBOW and Skip-Gram introduced from Word2Vec, which all can be optimized by negative sampling (NEG) and Hierarchical Softmax [1,2]. In terms of training efficiency, CBOW and negative sampling outperforms Skip-Gram and Hierarchical Softmax. Thus, the CBOW using NEG is applied in our model.

For current vertex v, we define its context vertices as $Context(v)$, the negative sampling procedure is defined as $NEG(v)$. Here, $NEG(v) \neq \phi$. For $\forall u \in D$, D is the vertex set. We define the label as follows:

$$L^v(u) = \begin{cases} 1, u = v \\ 0, u \neq v \end{cases} \tag{3}$$

where $L^v(u)$ is the sampling label of vertex u, for example, the label of positive sampling is 1, and the label of negative sampling is 0.

For one vertex and its contexts $(v, Context(v))$, our aim was to maximize the following probability in the entire representation space:

$$g(v) = \prod_{\xi\in\{v\}\cup NEG(v)} p(\xi\big|Context(v)) \tag{4}$$

where

$$p(\xi|Context(v)) = \begin{cases} \sigma(\boldsymbol{x}_v^T\boldsymbol{\theta}^\xi) & L^v(\xi) = 1 \\ 1 - \sigma(\boldsymbol{x}_v^T\boldsymbol{\theta}^\xi) & L^v(\xi) = 0 \end{cases}, \tag{5}$$

and $\sigma(x)$ is the Sigmoid function, $\boldsymbol{x}_v$ is the summing operation of each representation in $Context(v)$, $\boldsymbol{\theta}^\xi$ is an assistant training vector of the current vertex ξ. Equation (5) can also be formalized as follows:

$$p(\xi|Context(v)) = [\sigma(\boldsymbol{x}_v^T\boldsymbol{\theta}^\xi)]^{L^v(\xi)} \cdot [1 - \sigma(\boldsymbol{x}_v^T\boldsymbol{\theta}^\xi)]^{1-L^v(\xi)} \tag{6}$$

so, $g(v)$ can be calculated based on Equations (4) and (6) as follows:

$$g(v) = \prod_{\xi\in\{v\}\cup NEG(v)} [\sigma(\boldsymbol{x}_v^T\boldsymbol{\theta}^\xi)]^{L^v(\xi)} [1 - \sigma(\boldsymbol{x}_v^T\boldsymbol{\theta}^\xi)]^{1-L^v(\xi)}. \tag{7}$$

Therefore, for a given corpus C, the function:

$$G = \prod_{v\in C} g(v) \tag{8}$$

is defined as the overall objective function, the objective function of CBOW using NEG is denoted based on Equations (6)–(8) as follows:

$$\begin{aligned}\mathcal{L}(v) &= \log G = \log \prod_{v \in C} g(v) = \sum_{v \in C} \log g(v) \\ &= \sum_{v \in C} \sum_{\xi \in \{v\} \cup NEG(v)} \left\{ L^v(\xi) \cdot \log[\sigma(\boldsymbol{x}_v^T \boldsymbol{\theta}^\xi)] + (1 - L^v(\xi)) \cdot \log[1 - \sigma(\boldsymbol{x}_v^T \boldsymbol{\theta}^\xi)] \right\}.\end{aligned} \tag{9}$$

We optimize Equation (9) by the stochastic gradient ascent algorithm. Consequently, the updating of θ^u is computed as follows:

$$\boldsymbol{\theta}^\xi := \boldsymbol{\theta}^\xi + \mu \cdot [L^v(\xi) - \sigma(\boldsymbol{x}_v^T \boldsymbol{\theta}^\xi)] \boldsymbol{x}_v \tag{10}$$

The updating approach of each context vertex $v(u)$ is calculated as follows:

$$v(u) := v(u) + \mu \sum_{\xi \in \{v\} \cup NEG(v)} [L^v(\xi) - \sigma(\boldsymbol{x}_v^T \boldsymbol{\theta}^\xi)] \cdot \boldsymbol{\theta}^\xi \tag{11}$$

where $u \in Context(v)$.

3.5. MRNR Model

On citation networks, the relationships between vertices have reference relationships and referenced relationships. Generally, we just considered these relationships as a kind of edge existence. Some network representation learning algorithms based on shadow neural networks focus on learning the context relationships between the current vertex and context vertices. Consequently, the learnt representations tend to show the similarity if two vertices almost possess the same structure. Network vertices usually contain rich text contents on citation networks, such as paper title, abstract, and running title, etc. Empirically, the vertex without reference relationships maybe contain the same words in their titles, so we can construct unitary relationships and binary relationships by extracting one same word or two same words in titles. Therefore, we propose an optimized network representation learning algorithm using multi-relational data (MRNR) inspired by, literature [32,33], which jointly learns the network representations from the network representation learning model and knowledge representation learning model using TransE. Roughly speaking, MRNR integrates multi-relational data (triplets) into the procedure of learning the network representations. The detailed training diagram is shown in Figure 1.

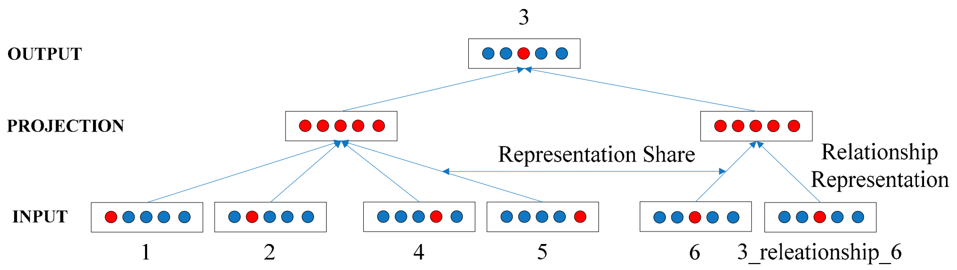

Figure 1. Learning framework of the proposed algorithm in this paper.

As shown in Figure 1, the left training procedures are based on CBOW modeling and the right training procedures are based on multi-relation data modeling. For the current vertex 3, its context vertices are 1, 2, 4, and 5, in addition, there exists some unitary relationships and binary relationships between vertex 3 and 6. For all vertices, two kinds of learning models share common representation space, for the relationships between vertices, MRNR creates a new representation space, which can avoid conflicts between vertex representations and relationship representations.

Figure 1 shows the detailed model training procedures and the following contents would introduce the objective function of MRNR.

Equation (9) is the objective function of DeepWalk using CBOW and negative Sampling, it then can be simplified as follows:

$$\mathcal{L}(v) = \sum_{v \in C} \log g(x), \tag{12}$$

The MRNR algorithm introduces the triplets (h, l, v) into DeepWalk, meanwhile, triplets (h, l, v) should satisfy $\boldsymbol{h} + \boldsymbol{l} \approx \boldsymbol{v}$ which means that $\boldsymbol{v}$ should possess a nearest vector space distance with $\boldsymbol{h} + \boldsymbol{l}$, otherwise, $\boldsymbol{h} + \boldsymbol{l}$ should possess a further vector space distance with $\boldsymbol{v}$. We get inspiration from the TransE algorithm, namely, if (h, l, v) holds, then the representation of the vertex v should be closed to the representation of the vertex h plus the representation of the relationship l.

Motivated by the TransE model, we finally use the DeepWalk algorithm to model all the triplets. Consequently, we propose a new approach based on DeepWalk to learn the relationships of the triplet (h, l, v). The objective function is as follows:

$$\mathcal{L}(\boldsymbol{v}, \boldsymbol{l}) = \sum_{v \in C} \sum_{l \in R_v} \log g_{h+l}(\boldsymbol{v}), \tag{13}$$

where R_v is the all triplets containing v. We can build the relationship between CBOW modeling and multi-relational modeling. In the MRNR model, we can regard the triplet (h, l, v) as a regularization term constraint to DeepWalk. To propose a joint learning framework, we define our objective function as follows:

$$\mathcal{L} = \mathcal{L}(v) + \alpha \mathcal{L}(v, l). \tag{14}$$

Here, we introduce the hyper-parameter α to balance the weights between structural features and triplet features.

For Equation (13), it can be replaced by Equation (9):

$$\begin{aligned} \mathcal{L}(v, l) = \sum_{v \in C} \sum_{l \in Rv} \sum_{\xi \in \{v\} \cup NEG(v)} & \Big\{ L^v(\xi) \cdot \log[\sigma(\boldsymbol{x}_{h+l}^{\mathrm{T}} \boldsymbol{\theta}^{\xi})] \\ & + (1 - L^v(\xi)) \cdot \log[1 - \sigma(\boldsymbol{x}_{h+l}^{\mathrm{T}} \boldsymbol{\theta}^{\xi})] \Big\} \end{aligned} \tag{15}$$

where $\boldsymbol{x}_{h+l} = \boldsymbol{x}_h + \boldsymbol{x}_l$. We use the stochastic gradient ascent algorithm to get the updating expressions of each parameter as follows:

$$\boldsymbol{\theta}^{\xi} := \boldsymbol{\theta}^{\xi} + \mu \cdot [L^v(\xi) - \sigma(\boldsymbol{x}_{h+l}^{\mathrm{T}} \boldsymbol{\theta}^{\xi})] \cdot \boldsymbol{x}_{h+l} \tag{16}$$

$$\boldsymbol{x}_h := \boldsymbol{x}_h + \mu \sum_{\xi \in (v\} \cup NEG(v)} [L^v(\xi) - \sigma(\boldsymbol{x}_{h+l}^{\mathrm{T}} \boldsymbol{\theta}^{\xi})] \cdot \boldsymbol{\theta}^{\xi} \tag{17}$$

$$\boldsymbol{x}_h := \boldsymbol{x}_h + \mu \sum_{\xi \in (v\} \cup NEG(v)} [L^v(\xi) - \sigma(\boldsymbol{x}_{h+l}^{\mathrm{T}} \boldsymbol{\theta}^{\xi})] \cdot \boldsymbol{\theta}^{\xi} \tag{18}$$

In Equation (14), we add the weight β for the triplet relation modeling. Therefore, we need to add a weight β to the Equations (16)–(18) when the variables are updated, namely, we replace the variable η with the variable $\eta \times \beta$.

3.6. Higher Order Proximity Approximation of MRNR

In Line [16] algorithm, the neighboring vertices are captured to provide the first order similarity and the second order similarity between vertices, so it provides two kinds of models, namely, 1st Line and 2nd Line. In addition, GraRep [17] adopts the form of adjacency matrix multiplication to capture higher order features. Moreover, WALKLETS [14] captures higher order vertex relationships on random walk sequences, then it puts all vertex relationships of lower order and higher order into the DeepWalk model, generating effective and excellent network representations. However, these algorithms possess a higher computation complexity. Yang et al. [13] propose a unified framework of higher order proximity approximation, NEU, which can approximate a higher order proximity matrix for learning better network representations. Importantly, NEU [13] applies arbitrary network representations trained by lower order proximities as inputs of NEU to avoid the repeated computation.

The optimization expression of NEU is as follows:

$$\boldsymbol{R}'_e = \boldsymbol{R}_e + \lambda_1 A \cdot \boldsymbol{R}_e + \lambda_2 A \cdot (A \cdot \boldsymbol{R}_e) \tag{19}$$

where $\boldsymbol{R}_e$ is the network representation vectors of lower order proximities, A is the normalized adjacency matrix, and $\boldsymbol{R}'_e$ is the network representation vectors of higher order proximity. Therefore, NEU can implicitly approximate higher order proximities based on a theoretical approximation bound. Therefore, we use the NEU algorithm to enhance the performance of MRNR, which is a kind of optimization for the learnt representation vectors of MRNR.

The algorithm framework proposed in this paper is not a task-specified algorithm for NLP and NRL tasks. Consequently, MRNR uses DeepWalk to verify the feasibility of the proposed framework. In network representation learning algorithms, some network representation learning algorithms are based on the framework of neural network, and others are based on matrix factorization. And MRNR uses the basic DeepWalk algorithm to construct a joint learning framework. Afterwards, some improved algorithms have been proposed based on the DeepWalk algorithm, such as node2vec [12], MMDW [18], and SDNE [19]. Therefore, MRNR is a flexible framework which also can be applied to other NRL tasks.

In order to demonstrate the modeling procedure of the MRNR algorithm in more detail, we give the algorithm pseudocode, which does not add the procedure of higher order optimization, the pseudocodes of Algorithm 1 are shown as follows:

Algorithm 1: MRNR Algorithm

Input: graph: $G(V, E)$, network representation vector size: k, random walk number: wn, random walk lengh: rl, triplet set: S, the weight of triplet: γ.
Output: vertex representation vectors of MRNR

```
// get all walk sequences of network vertices
1:  for $i$ form 1 to $wn$ do
2:      $O \leftarrow$ Shuffle($V$)
3:      for $v$ in $O$ do
4:          $C \leftarrow$ RandomWalkSequenceAppend($G, v, rl$)
5:      end for
6:  end for
    //initialize all variables
7:  $vertex\_amount \leftarrow$ GetVocabAmount($C$)
8:  $relation\_amount \leftarrow$ GetRaltionAmount($S$)
9:  $\boldsymbol{Vec} \leftarrow$ Init_Representation_Vector($vertex\_amount, k$)
10: $\boldsymbol{\theta} \leftarrow$ Init_Representation_Vector($vertex\_amount, k$)
11: $\boldsymbol{P} \leftarrow$ Init_Representation_Vector($relation\_amount, k$)
12: for $v$ in C do
13:     $e \leftarrow 0$
14:     $x_v \leftarrow \sum_{\xi \in Context(v)} \boldsymbol{Vec}(\xi)$
15:     for $\xi \in \{v\} \cup NEG(v)$ do
```

```
16:             $q \leftarrow \sigma(\boldsymbol{x}_v^{\mathrm{T}}\boldsymbol{\theta}^{\xi})$
17:                 $g \leftarrow \mu\cdot[L^v(\xi) - q]$
18:                 $e \leftarrow e + g\boldsymbol{\theta}^{\xi}$
19:             $\boldsymbol{\theta}^{\xi} \leftarrow \boldsymbol{\theta}^{\xi} + g\boldsymbol{x}_v$
20:         end for
21:         for $\xi$ in $Context(v)$ do
22:             $\boldsymbol{Vec}(\xi) \leftarrow \boldsymbol{Vec}(\xi) + e$
23:         end for
24:         for $l$ in $R_v$ do
25:             $e \leftarrow 0$
26:             $(h, l, v) \leftarrow$ ObtainTriplets($R$, $l$,$v$)
27:             for $\xi = \{v\} \cup NEG(v)$ do
28:                 $\boldsymbol{x}_{h+l} \leftarrow \boldsymbol{Vec}(h) + \boldsymbol{P}(l)$
29:                 $q \leftarrow \sigma(\boldsymbol{x}_{h+l}^{\mathrm{T}}\boldsymbol{\theta}^{\xi})$
30:                 $g \leftarrow \mu\cdot[L^v(\xi) - q]$
31:                 $e \leftarrow e + \mathrm{g}\boldsymbol{\theta}^{\xi}$
32:                 $\boldsymbol{\theta}^{\xi} \leftarrow \boldsymbol{\theta}^{\xi} + \gamma\ \mathrm{g}\ \boldsymbol{x}_{h+l}$
33:             end for
34:             $\boldsymbol{Vec}(v) \leftarrow \boldsymbol{Vec}(v) + \gamma$ e
35:             $\boldsymbol{P}(l) \leftarrow \boldsymbol{P}(l) + \gamma$ e
36:         end for
37:     end for
```

4. Experiments and Evaluations

4.1. Dataset Setup

We conducted our experiments on the Citeseer (M10), Database System and Logic Programming (DBLP) (V4), and Simplified DBLP (SDBLP) datasets. The descriptions of the three datasets are given in Table 2:

Table 2. Dataset descriptions.

Datasets	Original Node	Original Edge	Single Node	Used Node	Used Edge	Average Clustering Coefficient	Average Degree
Citeseer	10,310	5923	5700	4610	5923	0.264	2.57
DBLP	60,744	105,781	43,019	17,725	105,781	0.187	11.936
SDBLP	60,744	105,781	0	3119	39,516	0.259	25.339

The separate vertices are not connected with the other vertices. Therefore, the random walker cannot walk to the separate vertices for the DeepWalk, node2vec, and Line algorithms. For the GraRep algorithm, it was also impossible to learn the relationships between the separate vertices and other vertices, namely, the adjacency matrix was multiplied several times; however, the values in the multiplied matrix remained 0. Therefore, the NRL algorithm will get a poor performance for the sparse network containing separate vertices. Nevertheless, the NRL algorithm based on structures and vertex texts can gain satisfactory classification performance, such as the TADW algorithm. In order to fairly compare the performance of the above algorithms, we deleted all separate vertices on the Citeseer dataset as well as the DBLP dataset. The SDBLP dataset is a simplified dataset of DBLP. On the SDBLP, we only retained those vertices containing three edges at least.

The unitary and binary relationships between vertices were applied to the MRNR to improve the performance of DeepWalk. We used Equation (13) to model all the representations of triplets. The amount is shown in Table 3.

Table 3. Triplet and relationship count on the Citeseer, DBLP, and SDBLP.

Dateset	Citeseer		DBLP		SDBLP	
	Triplet	**Relationship**	**Triplet**	**Relationship**	**Triplet**	**Relationship**
Unitary Relationship	1,263,566	6524	15,848,328	11,858	510,793	4451
Binary Relationship	54,133	3470	115,540	9356	5799	1264

4.2. Baseline Algorithms

DeepWalk: DeepWalk can learn network representations for large-scale networks. It is a basic algorithm of NRL, which is usually applied as a baseline algorithm. It provides two kinds of training models, such as CBOW and Skip-Gram, and it also provides two kinds of optimization methods, such as Hierarchical Softmax and negative sampling. In our experiment, we used the CBOW and negative sampling for DeepWalk.

Line: Line is a network representation learning method proposed by Tang et al. [16]. And then Line studies the problem of encoding very large information networks into low-dimensional representation spaces. This method proposes the concept of first order proximity and second order proximity. Consequently, Line provides two kinds of models, 1st Line and 2nd Line. We use the 2nd Line model in this paper.

node2vec: Node2vec also adopts the BFS and DFS strategies to capture higher order neighboring vertices. Meanwhile, node2vec defines a second-order random walk to balance the DFS and BFS. The random walk path of node2vec is controlled by a parameter. Unlike the random walk of DeepWalk, the probability from the current vertex to next vertex is not exactly equal.

GraRep: GraRep learns the network representations by probability transition matrix, the probability transition matrix of the first order is $A = D^{-1}S$, where S is the adjacency matrix, and D is the diagonal matrix. The probability transition matrix of k-th order is A^k. In this paper, we set the k value to three, which means that GraRep could capture the neighboring vertices of the first order, second order, and third order.

Text Feature (TF): We took the vertex's text feature matrix $r_v \in \mathbb{R}^{|V| \times 100}$ as the 100 dimensional representations of the network vertices.

DeepWalk + TF: We concatenated two vectors from DeepWalk and text feature to obtain the 100 dimensional representations of network vertices.

DeepWalk as Matrix Factorization (MFDW): DeepWalk has proved that it factorizes the matrix $M = (A + A^2)/2$. Here, we used the singular value decomposition (SVD) [34] algorithm to factorize the matrix M. The network representation vectors are calculated by the expression $W = U \cdot S^{0.5}$.

TADW: The TADW incorporates text features of the networks into network representations with the IMC [7]. And it factorizes the matrix $M = (A + A^2)/2$, where A consists of the reciprocal of the degree value of the vertex.

Simplified TADW (STADW): Because TADW adopts normalization to optimize text features two times. The first optimization is conducted before SVD factorization of text features and the second optimization is performed after SVD factorization. In STADW, we deleted the optimization procedures for the text feature, because the MRNR proposed in this paper simply adopted the unitary and binary vertex relationships, which are not optimized by the normalization methods.

MRNR@1: MRNR uses the unitary relationships to optimize the performance of NRL.

MRNR@2: MRNR uses the binary relationships to optimize the performance of NRL.

MRNR@1 + NEU: the proposed MRNR uses the unitary relationships to optimize the performance of NRL, meanwhile, the learnt representations are optimized by NEU.

MRNR@2 + NEU: the proposed MRNR uses the binary relationships to the performance of NRL, meanwhile, the learnt representations are optimized by NEU.

4.3. Classifiers and Experiment Setup

We conducted our experiments on the Citeseer, DBLP, and SDBLP datasets. We used the network classification task to verify the performance of MRNR. The classifier uses SVM implemented by Liblinear [35]. In addition, we randomly took a portion of labeled data as a training set and the rest was a testing set. We evaluated the accuracy of the classification based on the different proportions of training sets. This proportion varied from 10% to 90%. For the above baseline algorithms, we set the dimension size as 100, walk length as 40, walk number as 10, window size as 5, negative number as 5, and minimum node frequency as 5. The minimum vertex frequency means that the algorithm will discard some vertices that appear less than five times in the random walk sequences. Note that the weight of the triplet was set to 0.5 in Section 4.4. We repeated our experiment 10 times and reported the average accuracy in the following experiments.

4.4. Experimental Results and Analysis

The classification results are shown in Table 4 (Citeseer), Table 5 (DBLP), and Table 6 (SDBLP). The MRNR consistently and significantly outperformed the most baseline algorithms on different training proportions, showing its feasibility on network vertex classification tasks.

Table 4. Accuracy (%) of vertex classification on the Citeseer dataset.

Labeled Nodes (%)	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**
DeepWalk	55.89	59.30	60.89	61.48	62.19	62.30	62.62	62.33	63.95
Line	42.64	47.06	48.04	49.57	50.43	51.02	51.18	53.07	53.63
node2vec	62.47	65.61	66.00	67.07	67.40	67.15	67.46	68.07	68.56
GraRep	39.38	53.09	57.85	59.75	59.97	61.05	61.57	62.09	60.89
Text Feature (TF)	57.69	61.30	62.76	63.05	63.48	63.30	62.87	62.19	63.95
DeepWalk + TF	58.31	61.15	62.73	63.37	64.18	63.96	65.50	65.49	65.30
MFDW	57.62	60.79	62.33	63.05	62.96	63.00	63.00	63.48	64.30
TADW	68.47	72.29	73.50	74.25	74.24	74.73	75.36	74.95	75.47
STADW	46.71	61.99	65.99	71.60	73.11	73.82	73.33	74.27	74.36
MRNR@1	76.31	77.85	78.33	78.43	79.31	79.39	79.37	79.64	79.72
MRNR@2	66.72	68.93	70.13	70.71	70.83	70.94	71.61	71.91	71.50
MRNR@1 + NEU	77.05	78.54	79.39	79.42	79.14	79.59	79.69	80.31	79.86
MRNR@2 + NEU	67.41	69.02	70.14	71.06	71.77	71.88	72.18	73.53	71.76

Table 5. Accuracy (%) of vertex classification on the DBLP dataset.

Labeled Nodes (%)	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**
DeepWalk	62.26	64.34	65.42	65.98	66.24	66.18	66.60	67.03	66.77
Line	64.49	66.53	67.49	67.87	67.98	68.30	69.03	68.89	68.86
node2vec	73.39	73.98	75.25	75.61	75.70	75.85	75.79	75.73	76.36
GraRep	58.90	65.90	67.26	67.92	68.77	68.88	69.26	69.56	69.79
Text Feature (TF)	66.17	69.46	70.49	71.15	71.29	71.44	71.54	71.57	71.83
DeepWalk + TF	62.61	65.15	65.99	66.22	66.37	66.60	67.03	66.91	67.61
MFDW	65.02	74.68	74.88	75.02	75.05	75.13	75.22	74.57	75.51
TADW	80.22	81.41	82.05	82.36	82.77	82.88	82.71	83.10	82.97
STADW	75.39	80.51	81.09	81.00	81.64	81.70	81.71	81.69	81.34
MRNR@1	82.61	83.91	84.39	84.72	84.65	84.92	85.01	85.46	85.87
MRNR@2	77.33	79.76	80.63	80.98	81.11	81.68	81.69	81.81	81.58
MRNR@1 + NEU	83.86	85.78	86.44	86.28	86.56	86.52	86.81	86.95	86.79
MRNR@2 + NEU	80.34	82.29	82.78	83.13	83.54	83.76	83.68	84.01	84.88

Table 6. Accuracy (%) of vertex classification on the SDBLP dataset.

Labeled Nodes (%)	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**
DeepWalk	79.76	80.65	81.88	81.49	82.56	82.35	82.73	82.71	83.37
Line	73.79	77.01	78.11	78.28	79.31	78.97	79.63	78.82	78.77
node2vec	82.17	82.87	83.09	84.51	84.45	84.01	84.84	84.73	84.98
GraRep	80.99	82.52	84.14	84.78	84.97	84.17	85.36	85.27	84.95
Text Feature (TF)	65.03	71.23	72.64	73.86	74.54	75.07	75.14	76.00	75.33
DeepWalk + TF	79.39	80.95	81.60	81.44	82.55	82.22	82.10	82.58	83.15
MFDW	79.79	83.08	84.38	84.12	84.53	84.29	84.70	84.55	84.53
TADW	81.38	83.13	83.54	84.84	84.09	84.34	85.00	85.69	85.78
STADW	80.63	81.92	82.54	83.60	84.02	84.92	85.03	84.65	85.27
MRNR@1	83.92	84.79	85.15	85.30	85.43	85.89	85.32	86.51	85.21
MRNR@2	83.80	84.61	85.29	85.41	85.32	85.34	86.03	85.15	85.46
MRNR@1 + NEU	83.84	84.88	85.18	85.58	85.75	86.35	85.73	85.68	85.85
MRNR@2 + NEU	83.44	84.74	84.87	85.01	85.29	85.36	85.61	85.71	85.11

From Tables 4–6, we derived the following interesting observations:

(1) the introduced MFDW is a matrix factorization form of the DeepWalk algorithm. The performance of the MFDW algorithm was better than that of DeepWalk for the three datasets. The performance of TF was better than that of DeepWalk for the Citeseer and DBLP datasets. In addition, the performance of TF was better than that of Line, achieving an almost same performance with DeepWalk on the SDBLP dataset. The main reason why MFDW can achieve a better performance is that all vertices in MFDW participate and construct the structural feature matrix. Nevertheless, the DeepWalk algorithm avoids accurately calculating the structure feature matrix. Because TADW is an algorithm based on the MFDW, which incorporates the vertex texts into network representations, therefore, TADW can obtain a better performance and outperform the other baseline algorithms. In addition, TADW uses the inductive matrix completion algorithm rather than a simple SVD factorization. Therefore, the performance of inductive matrix completion can greatly prompt the performance of TADW to a certain extent. The performance of DeepWalk + TF shows that it is difficult to obtain performance improvement by concatenating two representations from DeepWalk and text features. In addition, the experimental results have proved that modeling the triplet relationships generally improves the performance of the original DeepWalk. Indeed, the proposed learning framework in this paper can be applied to other types of models.

(2) The performance of the MRNR@1 was better than that of TADW, DeepWalk, node2vec, MFDW, Line, and TF on the Citeseer, DBLP, and SDBLP. The performance of the MRNR@2 was slightly inferior than TADW on the Citeseer and DBLP. However, the performance of the MRNR@1 and MRNR@2 was superior to that of TADW on the SDBLP. The average degrees of Citeseer, DBLP, and SDBLP were 2.57, 11.936, and 25.339. The mean differences between MRNR@1 and TADW were 4.64%, 2.33%, and 1.6% on the Citeseer, DBLP, and SDBLP. It demonstrates that the performance differences were decreasing with the growth of the average degree. On the SBDLP dataset, although all baseline algorithms almost had the same performance, MRNR@1 and MRNR@2 achieved the best performance.

(3) The classification performance of the MRNR@1 was better than that of the MRNR@2, which demonstrated that the more triplets between vertices, the better the performance of the MRNR. However, the number of triplets had little impact on the performance of network representation learning on the dense SDBLP dataset. In addition, the learnt network representations were optimized by the NEU algorithm, therefore, the NRL performance of the MRNR@1 and MRNR@2 achieved significant improvement on the Citeseer and DBLP datasets. On the DBLP dataset, the optimization effect of NEU was not as good as that on the Citeseer and DBLP.

(4) the introduced TADW algorithm and the proposed MRNR algorithm also used text features to enhance the performance of network representation learning. More specifically, TADW used matrix factorization to incorporate the text features into network representations, but the MRNR adopted vertex triplets constructed by text features to constrain the modeling procedure of the MRNR. In the

MRNR algorithm, we set the minimum vertex frequency to 5, which means these vertices with the degree value of less than 5 will be deleted in the procedure of random walk sampling. The influence of this behavior is that vertices of small occurrence frequency are not be input into the model to learn as the context vertices, which are only trained when they are the current vertices. In the TADW algorithm, all the vertices take part in the training procedure. The purpose of setting the minimum vertex frequency is to speed up the convergence of the model. Therefore, deleting low-frequency vertices in contexts can help improve the performance of network representation learning. However, STADW is a simplified algorithm based on TADW, which deletes all optimizations of text features, consequently, the performance of STADW is inferior to that of TADW on three datasets.

4.5. Parameter Sensitivity

The proposed MRNR algorithm converts the texts of the network vertices into the triplet form, and then MRNR concatenates the objective of the triplet constraint to the that of DeepWalk. In order to research the effect of the triplet on NRL, the weight parameter α was added to the constraint item of the triplet in Equation (14). By setting a weight value that varies from 0.1 to 0.9, we counted the accuracy of the classification tasks under different proportions of training datasets that ranged from 10% to 90% for each fixed weight value, and we then calculated the average accuracy in the last column. Because the classification performance of the triplets consisted of unitary relationships, it was better than that of the triplets composed of binary relationships; therefore, we only considered the triplets of binary relationships in this experiment. The detailed results are shown in Tables 7–9.

Table 7. Influence of triplet weight on the Citeseer dataset.

α	**The Proportions of Training Datasets**									**Average**
	10%	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	
0.1	60.67	63.66	64.97	65.23	66.12	66.68	66.90	66.89	67.09	65.36
0.2	64.16	66.51	68.15	68.63	68.88	68.87	68.67	69.12	70.30	68.14
0.3	64.70	69.03	69.56	69.73	70.10	71.26	70.60	70.72	71.43	69.68
0.4	66.50	68.72	69.70	70.37	70.87	70.99	70.91	71.63	71.43	70.12
0.5	66.72	68.93	70.13	70.71	70.83	70.94	71.61	71.91	71.50	70.36
0.6	67.22	69.21	70.31	70.59	71.28	71.25	71.97	71.53	72.07	70.60
0.7	66.86	69.41	70.16	70.83	71.46	72.15	72.09	72.31	73.22	70.94
0.8	66.72	69.80	70.53	71.27	71.72	72.32	72.70	73.19	73.30	71.28
0.9	67.28	69.31	70.81	71.71	72.07	72.15	72.97	72.25	72.72	71.25

Table 8. Influence of triplet weight on the DBLP dataset.

α	**The Proportions of Training Dataset**									**Average**
	10%	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	
0.1	77.98	79.58	80.31	80.62	80.94	81.08	81.44	81.28	81.16	80.49
0.2	77.62	79.45	80.44	80.74	81.05	81.00	81.07	80.95	81.39	80.41
0.3	77.23	79.80	80.47	80.90	80.88	81.25	81.16	81.47	82.25	80.60
0.4	77.52	80.04	80.47	80.87	81.29	81.70	81.45	81.61	81.90	80.76
0.5	77.33	79.76	80.63	80.98	81.11	81.68	81.69	81.81	81.58	80.73
0.6	77.49	79.75	80.61	80.89	81.18	81.49	81.87	81.87	82.08	80.80
0.7	77.57	79.94	80.75	81.06	81.36	81.70	81.68	81.94	81.81	80.87
0.8	77.55	79.91	80.61	80.96	81.15	81.45	81.65	81.96	81.96	80.80
0.9	78.06	80.04	80.78	80.95	81.29	81.51	81.77	81.99	82.34	80.97

Table 9. Influence of triplet weight on the SDBLP dataset.

α	The Proportions of Training Dataset									Average
	10%	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	
0.1	83.74	84.62	85.52	85.37	85.44	85.31	85.75	85.60	85.92	85.25
0.2	83.97	84.97	84.90	85.02	84.96	85.77	85.48	85.86	85.76	85.19
0.3	83.82	85.16	84.96	85.29	85.54	85.43	85.67	85.33	86.85	85.34
0.4	83.84	84.41	85.33	84.93	85.21	85.47	86.19	85.07	86.27	85.19
0.5	83.80	84.61	85.29	85.41	85.32	85.34	86.03	85.15	85.46	85.15
0.6	83.94	84.87	85.10	85.39	85.45	85.65	86.28	85.62	85.50	85.31
0.7	83.94	84.75	85.03	85.03	84.93	85.70	85.54	85.67	85.02	85.07
0.8	83.78	84.63	85.29	85.37	85.52	85.55	85.85	85.65	86.53	85.35
0.9	83.93	84.71	85.03	85.03	85.50	85.31	85.48	86.21	85.56	85.20

As shown in Tables 7–9, the classification accuracy slightly varied within 1%, with the increasing weight of the triplet on the DBLP and SDBLP datasets. However, the fluctuation of classification accuracy was relatively clear on the Citeseer dataset. Compared with the triplet weight of 0.1, the classification accuracy of the triplet weight of 0.9 approximately improved by 5.89%. In addition, the Citeseer dataset was a sparse network, and the DBLP and SDBLP datasets were dense networks. In conclusion, we found that the weight of the triplet had a great influence on the representation performance of sparse network representation learning, but the weight of the triplet had little effect on the representation performance of the dense network representation learning. The main reason was that the sufficient structural features enable all kinds of algorithms to almost possess the same performance on dense networks. On sparse networks, the opposite was true.

4.6. Visualizations

In the previous sections, we used classification tasks to verify the performance of the MRNR on the Citeseer, DBLP, and SDBLP. In this section, we explore the clustering ability of the learnt representations. We first randomly chose four network categories on the Citeseer and three categories on the DBLP and SDBLP. Each category contained 150 vertices on the Citeseer and 200 vertices on the DBLP and SDBLP. We used the t-SNE [36] algorithm to visualize the learnt representation vectors. The visualization results are shown in Figure 2.

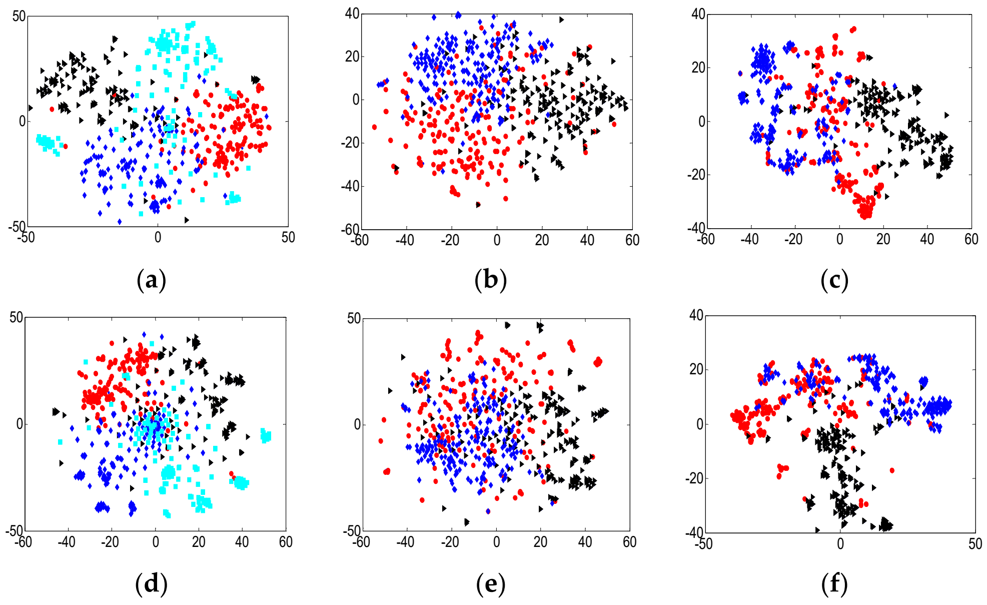

Figure 2. Visualization experiments. (**a**–**c**) The MRNR@1′s visualization results on the Citeseer, DBLP, and SDBLP. (**d**–**f**) The MRNR@2′s visualization results on the Citeseer, DBLP, and SDBLP.

As shown in Table 2, the amount of triplets constructed by the unitary relations was more than that of the triplets constructed by binary relations, meanwhile, the amount of unitary relations was more than that of the binary relations on the Citeseer, DBLP, and SDBLP. Consequently, the visualization results show that the clustering ability of MRNR@1 is superior to that of MRNR@2 on the Citeseer and DBLP, and they almost achieved the same clustering performance on the SDBLP. The visualization experiments show that the number of triplets would affect the visualization and clustering performance on the sparse networks, namely, enough triplets would contribute to the feature learning. Consequently, we can obtain excellent representation vectors if the massive triplets are introduced into network representation learning on the sparse networks. On the dense networks, the influence of the number of triplets on visualization is negligible. Moreover, the results of visualization are just consistent with the results of classification in Tables 4–6. Therefore, the feasibility of the proposed MRNR algorithm is confirmed from these two experiments.

4.7. Case Study

The proposed MRNR can learn the network representations using multi-relational data. In this section, we take a vertex's title as an example to conduct an instance study on the Citeseer dataset. The target title is "An Evolutionary Algorithm That Constructs Recurrent Neural Networks". Based on DeepWalk, TADW, MRNR@1, and MRNR@2, we then found the most relevant vertices using the cosine similarity method. We obtained the top three nearest neighboring vertices in Table 10. The triplet weight was set as 0.5 in this experiment.

Table 10. Case study on the Citeseer dataset.

Algorithm	Title	Similarity
DeepWalk	The Evolution of Communication in Adaptive Agents	0.8274
	Challenges and Opportunities of Evolutionary Robotics	0.8031
	Applying Evolutionary Computation to Designing Neural Networks: A Study of the State of the Art	0.7998
TADW	Pareto Evolutionary Neural Networks	0.7705
	Compensating for Neural Transmission Delay using Extrapolatory Neural Activation in Evolutionary Neural Networks	0.7491
	An Adaptive Merging and Growing Algorithm for Designing Artificial Neural Networks	0.7449
MRNR@1	Applying Evolutionary Computation to Designing Neural Networks A Study of The State of The Art	0.9508
	Making Use of Population Information in Evolutionary Artificial Neural Networks	0.9447
	Competitive Coevolution Through Evolutionary Complexification	0.9415
MRNR@2	On the Combination of Local and Evolutionary Search for Training Recurrent Neural Networks	0.9736
	Applying Evolutionary Computation to Designing Neural Networks a Study of The State of The Art	0.9648
	Evolutionary Reinforcement Learning of Artificial Neural Networks	0.9647

As shown in Table 10, TADW uses a text feature matrix to improve the performance of network representation learning and MRNR applies the knowledge-based triplets to improve the performance of network representation learning, consequently, both TADW and MRNR show text generality and similarity based on the learnt representations. For example, the titles generated by TADW, MRNR@1, and MRNR@2 contain the same word "Neural Networks" or "Evolutionary". This kind of representations trained by MRNR can be used for various tasks, such as recommendation systems, webpage ranking, etc. This example demonstrates that TADW and MRNR learn the network representations by balancing the weight between the structural feature similarity and text similarity.

5. Conclusions

In this paper, we introduced the idea of knowledge representation learning into the network representation learning, which is that we used knowledge triplets to constrain the training procedure of network representation learning. Consequently, we proposed a new network representation learning algorithm, MRNR, which uses the vertex's texts to construct unitary and binary triplets between vertices. To testify the performance of MRNR, we conducted classification, visualization, and case study experiments on the Citeseer, DBLP, and SDBLP datasets. We found that the greater the number of triplets between vertices, the higher the accuracy of network classification, and the same results were obtained in visualization tasks. In this paper, we specially compare the classification performance of MRNR with the TADW algorithm. We found that the classification performance of TADW was inferior to that of the MRNR algorithm on the Citeseer, DBLP, and SDBLP. The MRNR and TADW also adopted text features to improve the performance of network representation learning. The difference was that TADW converts text features into a text matrix and MRNR converts the text features into knowledge-based triplets. Finally, according to the application requirements, we should choose the appropriate algorithm between MRNR and TADW. The MRNR algorithm proposed in this paper is more suitable for the task of large-scale network representation learning using text features due to its effectiveness. For small-scale networks, the matrix factorization-based TADW algorithm is also a suitable approach.

Author Contributions: Conceptualization, Z.Y., K.Z. and Y.Z.; Methodology, Z.Y., H.Z.; Software, K.Z., Z.W.; Validation, Z.Y. and Y.Z.; Formal Analysis, Y.Z.; Investigation, Z.Y.; Resources, Z.Y.; Data Curation, K.Z.; Writing—Original Draft Preparation, Z.Y.; Writing—Review and Editing, Y.Z. and Z.W.; Visualization, Z.Y.; Project Administration, Z.Y.; Funding Acquisition, Z.Y.

Funding: This research was supported by the Fundamental Research Funds for the Central Universities, China (Grant Number: 2017TS045).

Acknowledgments: The authors wish to express their gratitude to the anonymous reviewers and the associate editors for their rigorous comments during the review process. In addition, the authors also would like to thank the experimenters in our laboratory for their great contribution toward data collection; they are Ke Zhang, Zhaoyang Wang and Yu Zhu.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. In Proceedings of the ICLR, Scottsdale, AZ, USA, 2–4 May 2013.
2. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the NIPS, Las Vegas, USA, 9–13 December 2013; pp. 3111–3119.
3. Perozzi, B.; Al-Rfou, R.; Skiena, S. Deepwalk: Online learning of social representations. In Proceedings of the KDD, New York, NY, USA, 24–37 August 2014.
4. Li, G.H.; Luo, J.W.; Xiao, Q. Predicting microRNA-disease associations using network topological similarity based on deepWalk. *IEEE Access* **2017**, *5*, 24032–24039. [CrossRef]
5. Yang, C.; Liu, Z.Y.; Zhao, D.L.; Sun, M.S. Network representation learning with rich text information. In Proceedings of the AAAI, Austin, TX, USA, 25–30 January 2015.
6. Pan, S.R.; Wu, J.; Zhu, X.Q.; Wang, Y. Tri-party deep network representation. In Proceedings of the IJCAI, New York, NY, USA, 9–15 July 2016.
7. Natarajan, N.; Dhillon, I.S. Inductive matrix completion for predicting gene-disease associations. *Bioinformatics* **2014**, *30*, 60–68. [CrossRef] [PubMed]
8. Bordes, A.; Usunier, N.; Garcia-Duran, A. Translating embeddings for modeling multi-relational data. In Proceedings of the NIPS, Harrah, NV, USA, 28 March–2 September 2013.

9. Bollacker, K.; Evans, C.; Paritosh, P.; Taylor, J. Freebase: A collaboratively created graph database for structuring human knowledge. In Proceedings of the SIGMOD, Vancouver, BC, Canada, 9–12 June 2008.
10. Lehmann, J. DBpedia: A large-scale, multilingual knowledge base extracted from wikipedia. *Semant. Web* **2015**, *6*, 167–195.
11. Bengio, Y.; Courville, A.; Vincent, P. Representation learning: A review and new perspectives. *IEEE Trans. Pami* **2013**, *35*, 1798–1828. [CrossRef] [PubMed]
12. Grover, A.; Leskovec, J. node2vec: Scalable feature learning for networks. In Proceedings of the SIGKDD, San Francisco, CA, USA, 26 June–1 July 2016.
13. Yang, C.; Sun, M.S.; Liu, Z.Y.; Tu, C.C. Fast network embedding enhancement via high order proximity approximation. In Proceedings of the IJCAI, Melbourne, Australia, 19–25 August 2017.
14. Perozzi, B.; Kulkarni, V.; Skiena, H.C.; Skiena, S. Don't walk, skip! online learning of multi-scale network embeddings. *arXiv* **2017**, arXiv:1605.02115.
15. Chen, H.C.; Perozzi, B.; Hu, Y.F.; Skiena, S. HARP: Hierarchical representation learning for networks. In Proceedings of the AAAI, New Orleans, LA, USA, 2–7 February 2018.
16. Tang, J.; Qu, M.; Wang, M.Z.; Zhang, M.; Yan, J.; Mei, Q.Z. Line: Large-scale information network embedding. In Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18–22 May 2015.
17. Cao, S.S.; Lu, W.; Xu, Q.K. GraRep. Learning graph representations with global structural information. In Proceedings of the CIKM, Melbourne, VIC, Australia, 19–23 October 2015.
18. Tu, C.C.; Zhang, W.C.; Liu, Z.Y.; Sun, M.S. Max-Margin DeepWalk: Discriminative Learning of Network Representation. In Proceedings of the IJCAI, New York, NY, USA, 9–15 July 2016.
19. Li, J.Z.; Zhu, J.; Zhang, B. Discriminative deep random walk for network classification. In Proceedings of the ACL, Berlin, Germany, 7–12 August 2016.
20. Kipf, T.N.; Welling, M. Semisupervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.
21. Wang, X.; Cui, P.; Wang, J. Community Preserving Network Embedding. In Proceedings of the AAAI, San Francisco, CA, USA, 4–9 February 2017.
22. Huang, X.; Li, J.D.; Hu, X. Accelerated attributed network embedding. In Proceedings of the SDM, Houston, TX, USA, 27–29 April 2017.
23. Zhang, Y.; Lyu, T.S.; Zhang, Y. COSINE: Community-preserving social network embedding from Information diffusion cascades. In Proceedings of the AAAI, New Orleans, LA, USA, 2–7 February 2018.
24. Cavallari, S.; Zheng, V.W.; Cai, H.Y.; Chang, K.C.C.; Cambria, E. Learning community embedding with community detection and node embedding on Graphs. In Proceedings of the CIKM, Singapore, 6–10 November 2017.
25. Zhou, L.K.; Yang, Y.; Ren, X.; Wu, F.; Zhuang, Y.T. Dynamic network embedding by modeling triadic closure process. In Proceedings of the AAAI, New Orleans, LA, USA, 2–7 February 2018.
26. Ma, J.X.; Cui, P.; Zhu, W.W. Depth LGP: Learning embeddings of out-of-sample nodes in dynamic networks. In Proceedings of the AAAI, New Orleans, LA, USA, 2–7 February 2018.
27. Tu, K.; Cui, P.; Wang, X.; Wang, F.; Zhu, W.W. Structural deep embedding for hyper-networks. In Proceedings of the AAAI, New Orleans, LA, USA, 2–7 February 2018.
28. Goodfellow, I.L.; Pouget-Abadie, J.K.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks. In Proceedings of the NIPS, Palais des Congrès de Montréal, Montréal, QC, Canada, 8–13 December 2014.
29. Dai, Q.Y.; Li, Q.; Tang, J.; Wang, D. Adversarial network embedding. In Proceedings of the AAAI, New Orleans, LA, USA, 2–7 February 2018.
30. Wang, H.W.; Wang, J.; Wang, J.L.; Zhao, M.; Zhang, W.N.; Zhang, F.Z.; Xing, X.; Guo, M.Y. GraphGAN: Graph Representation Learning with Generative Adversarial Nets. In Proceedings of the AAAI, New Orleans, LA, USA, 2–7 February 2018.
31. Bojchevski, A.; Shchur, O.; Zügner, D.; Günnemann, S. NetGAN: Generating graphs via random walks. In Proceedings of the ICML, Long Beach, CA, USA, 10–15 June 2018.
32. Liu, Q.; Jiang, H.; Wei, S.; Ling, Z.H.; Hu, Y. Learning Semantic Word Embeddings based on Ordinal Knowledge Constraints. In Proceedings of the ACL, Beijing, China, 26–31 July 2015.
33. Nguyen, K.A.; Walde, S.S.I.; Vu, N.T. Integrating Distributional Lexical Contrast into Word Embeddings for Antonym-Synonym Distinction. In Proceedings of the ACL, Berlin, Germany, 8–12 August 2016.

34. Hofmann, T. Probabilistic latent semantic indexing. *ACM SIGIR* **1999**, *51*, 50–57.
35. Fan, R.E.; Chang, K.W.; Hsieh, C.J. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.* **2008**, *9*, 1871–1874.
36. Maaten, L.V.D.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.