Network Representation Learning Based on Topological Structure and Vertex Attributes

Shengxiang Hu, Bofeng Zhang(✉), Ying Lv, Furong Chang, and Zhuocheng Zhou

School of Computer Engineering and Science, Shanghai University, Shanghai, China
{mathripper,bfzhang}@shu.edu.cn

Abstract. Network Representation Learning (NRL) is an essential task in the field of network data analysis, which tries to learn the distributed representation of each vertex in the network for downstream vector-based data mining tasks. NRL is helpful in solving the computationally expensive or intractable problems of large-scale network analysis. Most related NRL methods only focus on encoding the network topology information into vertex representation. However, vertices may contain rich attributes that directly impact the network formation and measure the attribute-level similarity between vertices. Additionally, encoding the vertex attributes information into the representation vector may improve the performance of the representation. This paper proposes a general NRL framework TAFNE that can effectively retain both network topology and vertex attributes information. For complex types of vertex attributes, we design two different information fusion methods that take both training efficiency and generality into account. The proposed TAFNE framework is extensively evaluated through various data analysis tasks, including clustering, visualization and node classification, and achieves superior performance compared with baseline methods.

Keywords: Network Representation Learning · Vertex attributes · Network topology · Information fusion

1 Introduction

Deep Learning has achieved great success in analyzing Euclidean data such as natural languages, images, and audio. However, the non-Euclidean data is also valuable in daily life, effectively analyzing such data to extract favorable information seems to be a challenge. Many network-based data mining tasks, e.g. node classification [21], link prediction [10], recommendation [27], and key user discovery [7] are applied in various fields. For example, in the biological protein network, link prediction task is applied to study the correlation between proteins.

Supported by National Key R&D Program of China grant (NO. 2017YFC0907505).

Many studies used a discrete matrix to represent network data and perform network analysis tasks based on matrix spectral decomposition [12]. However, such methods have at least a quadratic time complexity respect to the number of vertices, which makes them difficult to generalize to large networks. Besides, it is also a tedious task to design specific algorithms for different networks.

To solve such problems, NRL aims to learn low-dimensional potential representations of network vertices that encode the network topology, vertex attribute, and other related information. The learned representation vectors can be used in subsequent vector-based machine learning tasks [29]. Most of the NRL methods focus on how to preserve network topology, such as DeepWalk [17], LINE [23], Node2Vec [5] and DNE [22] etc. These methods expect to keep vertices with similar topological contexts adjacent to each other in the new low-dimensional representation space to retain network topology information. However, the vertices representations learned by these methods may loss some information about the original network. For example, in a citation network, each vertex represents a paper with the abstract as its attribute, edges refer to the citation relationship among vertices. Two papers may study similar problems but are not directly connected or have no common neighbors. Considering only topology information cannot clearly explain the similarity between them, but their attributes can. So, vertex attribute information should also be taken into account to augment the network representation performance.

This paper proposes a Topology and Vertex Attributes Fusion Network Embedding (TAFNE) which can effectively encode both network topology and vertex attribute information into representations. Firstly, we train a Transformer-decoder [25] to capture topology information from the random walk vertex sequences. In this way, we can improve feature extraction capabilities through the self-attention mechanism. Then we suggest two different information fusion methods to preserve vertex attributes. Vertex attributes do not have to be numerical or nominal, but can also be, for example, full-text descriptions. We evaluate our approach on three data mining tasks: node classification, clustering, and visualization on attributed and non-attributed networks. The experimental results show that TAFNE outperforms state-of-the-art models.

The major contributions of this paper are summarized as follows.

- To preserve the topology information, we design an embedding model that combines a network structure embedding layer with the Transformer-decoder to take advantage of its strong feature extraction capabilities.
- To preserve vertex attribute information, we propose two information infusion methods that make vertices with similar attributes adjacent to each other in the low dimensional representation space.
- To evaluate the proposed TAFNE framework, we have widely conducted node classification, clustering, and visualization tasks on different types of datasets and achieved excellent performance.

The rest of this paper is organized as follows: In Sect. 2, we summarize the related works. In Sect. 3, we explain the research question and some essential concepts. In Sect. 4, we introduce the TAFNE framework in detail. In Sect. 5, we evaluate the TAFNE framework through various data mining tasks and present the experimental results. Section 6, we summarize the work in this paper.

2 Related Work

According to the different implementations of the algorithms, we divide the existing NRL methods into two categories: matrix factorization based methods and neural network based methods.

The matrix factorization based methods use different types of matrices to preserve the network information, such as adjacency matrix, k-step transition probability matrix, and context matrix [28], then leverage matrix factorization to obtain the network representations. Spectrum Embedding [2] is a method for calculating non-linear embeddings, which uses a Laplacian spectral decomposition to find low-dimensional representations for the input network data. Maximize Modularity [24] performs a decomposition of modularity matrix to learn community-oriented vertex representation [15]; TADW [28] executes inductive matrix decomposition [14] on the vertex context matrix to retain both network structure and vertex text feature in the representation vector. However, due to the fact that matrix factorization demands a lot of memory and computing resources, these matrix factorization based methods are difficult to extend to large networks. It is no longer suitable for the current Internet environment. What is more, the design of relational matrices will directly affect the matrix factorization performance, which brings additional contingency.

The neural network based methods have been proposed in order to extract sophisticated structural features and learn highly non-linear vertex representation recently. Such methods can usually be well extended to large networks. DeepWalk [17] performs truncated random walk on the network to generate vertex sequence sets. The frequency of the occurrence of vertex context pairs reflects their relevance. It learns from the experience of word representation learning and introduces the word embedding algorithm (Skip-Gram) [13] to learn the vertex representation over the vertex sequences. In the Struc2vec [19] method, vertex structural role proximity is encoded into a multilayer graph, and then DeepWalk is performed on the multilayer graph to learn vertex representations. SDNE [26] uses first and second-order similarity to preserve the network structure and designs a deeply embedding model for capturing highly non-linear network structures while retaining global and local structure information. DNE [22] utilizes LSTM [4] to keep the transfer possibilities among the nodes and designs a Laplacian-supervised embedding space optimization to preserve network local structure information. Compared with the matrix factorization based methods, the neural network based methods are easy to generalize and more robust because they are not affected by artificially designed relation matrices. However, most of the above works consider only network topology information. In this paper, we aim to propose a general NRL framework that encodes both network topology and the rich types of vertex attributes into the representation.

3 Problem Definition

In this section, we define the research problem and some important concepts.

Definition 1 (Network). *The network is defined as* $G = \{V, E, A\}$*, where* $V = \{v_1, \ldots, v_n\}$ *is the vertex set of network* G *and* n *is the number of vertices.* $E = \{e_{ij}\}_{i,j=1}^{n}$ *is the edge set of network* G*,* e_{ij} *denotes the edge from vertex* v_i *to* v_j*, which is attached with a weight* $w_{ij} \in \mathbb{R}$*,* $w_{ij} = 1$ *in unweighted network or* $w_{ij} = 0$ *if vertex* v_i*,* v_j *are not linked directly.* $A = \{\boldsymbol{a_1}, \ldots, \boldsymbol{a_n}\}$ *denotes the attributes set of the vertices,* $\boldsymbol{a_i}$ *means the attributes of vertex* v_i*.*

Definition 2 (Topology and Vertex Attributes Fusion Network Embedding). *Given a network* $G = \{V, E, A\}$*, the Topology and Vertex Attributes Fusion Network Embedding task aims to learn a representation matrix* $R \in \mathbb{R}^{|V| \times d}$*, where* $d \ll |V|$ *and the i_th row of* $R(R_i)$ *is the low-dimensional representation vector of vertex* v_i*. Meanwhile the representation vector* R_i *preserves both the context structural information and vertex attributes, which means the vertices with similar attributes or similar context structure in the source network are also adjacent to each other in the embedding space.*

4 Topology and Vertex Attributes Fusion Network Embedding

In this section, we present details of the proposed NRL framework TAFNE as shown in Fig. 1. The TAFNE framework consists of three parts: (a) A random walk process guided by second-order biased proximity, (b) A multi-layer

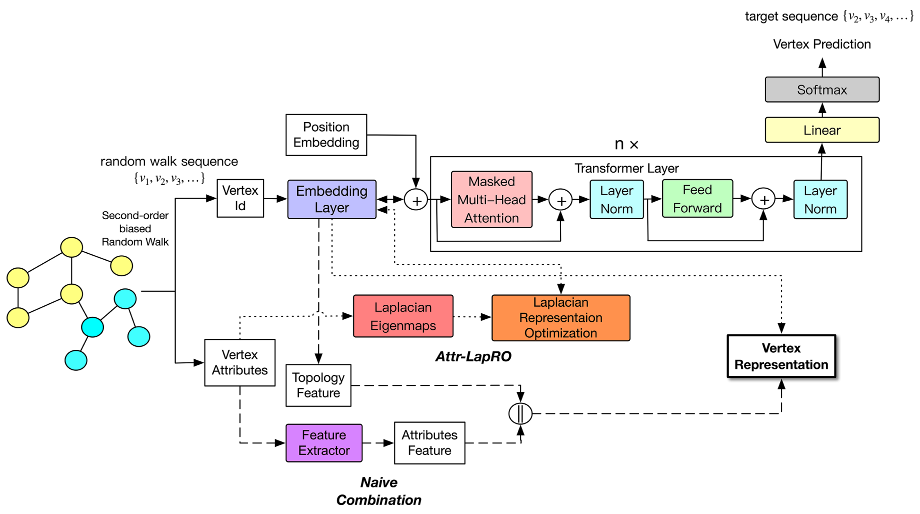


Fig. 1. The proposed Topology and Vertex Attributes Fusion Network Embedding (TAFNE) framework.

Transformer-decoder [25] for capturing network topology information preserved in the vertex sequences, (c) A information fusion model for preserving the vertex attributes.

4.1 Second-Order Biased Random Walk

In real networks, edge usually represents the similarity or some kinds of interactions between two vertices. Some related works, e.g., LINE [23], leverage the first-order similarity to guide the random walk process. The shortcoming is that, in unweighted networks, the degree of pairwise proximity and the social relationship information are left out. As shown in Fig. 2, since vertex i and j share more neighbors, i is closer with j than k in terms of social relationship. Here we utilize the second-order biased proximity to characterize the vertex similarity and guide the random walk process. For each pair of directly connected vertices, the similarity s_{ij} and the walk probability P are calculated as follows.

$$\begin{aligned} s_{ij} &= w_{ij}\frac{\left|N_{v_i} \bigcap N_{v_j}\right| + 1}{max(d_i, d_j)} \\ P(v_i \rightarrow v_j) &= \frac{e^{s_{ij}}}{\sum_{k=0}^{d_i} e^{s_{ik}}} \end{aligned} \tag{1}$$

where w_{ij} is the weight of edge e_{ij}, d_i is the degree of vertex v_i, N_{v_i} is the set of one-hop neighbors of vertex v_i.

4.2 Topology Information Preservation

The sequences generated by the random walk can be treated as sentences in natural language. The context information of the tokens in the sentences reflects the network's topology. The pairwise co-occur frequency of the vertices reflects the correlation between them. Following the idea of DeepWalk, we assume that the vertices with similar contexts (neighbors) are also similar, and are close to each other in the target embedding space. With such a hypothesis, we expect to maximize the likelihood of the central vertex when given contextual vertices. Given

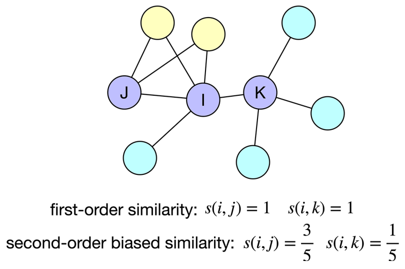

J
I
K
first-order similarity: $s(i,j) = 1$ $s(i,k) = 1$
second-order biased similarity: $s(i,j) = \frac{3}{5}$ $s(i,k) = \frac{1}{5}$

Fig. 2. A toy example of networks. First-order similarity and second-order biased similarity is quite different among vertices i, j and k.

a random walk sequence $S = \{v_1, \dots, v_n\}$, the likelihood function is defined as follows.

$$D(S) = \sum_i P(v_i \mid v_1, ..., v_{i-1}; W) \tag{2}$$

where W means the model parameters.

In related researches, DNE [22] chose LSTM as its prediction model, but LSTM cannot train in parallel and lack the ability to resolve long-distance dependencies. Since Transformer [25] proposed in 2017, Transformer and its variants [9,18] have achieved encouraging results on various natural language processing tasks. The self-attention mechanism can not only capture longer distance language structures but also train in parallel to speed up the training process. Therefore, we employ a multi-layer Transformer-decoder [25] as our prediction model to take advantage of its powerful feature extraction capabilities. The Transformer-decoder applies a masked multi-head self-attention operation, followed by a position-wise feedforward layer, over the input context vertices to produce an output distribution over target vertex. This process and the loss function L_{lm} can be formulated as follows:

$$\begin{aligned} h_0 &= T_S R^e + R^p \\ h_l &= transformer\_decoder(h_{l-1}) \ \forall l \in [1, n] \\ \hat{y} &= softmax(h_n) \\ L_{lm}(\hat{y}, y) &= -\sum y * log(\frac{1}{\hat{y}}) \end{aligned} \tag{3}$$

where $T_S \in \mathbb{R}^{|V| \times |V|}$ is the tokenized matrix of vertex sequence S, T_i is the one-hot vector for vertex v_i, $R^e \in \mathbb{R}^{|V| \times d_e}$ is the vertex embedding matrix, $R^p \in \mathbb{R}^{|V| \times d_e}$ is the position embedding matrix, n is the number of layers, y is the target predicted vertex. We only want to get the vertex embedding feature through the prediction process, so the output of the embedding layer $R^e \in \mathbb{R}^{|V| \times d_e}$ is what we expect.

4.3 Information Fusion

In this subsection, we present the details of the information fusion model, as shown in the lower part of Fig. 1, which is designed to encode vertex attributes information into vertex representation. We expect vertices with similar attributes to be adjacent to each other in the low-dimensional target embedding space as well. Because of the rich types of vertex attributes, we propose two ways to incorporate the features extracted from the attributes of network vertices into the embedding process, which guarantees that the proposed NRL framework can not only train more efficiently but also be applied to various kinds of networks.

Naive Combination. For textual attributes or numerical attributes with sufficient dimensions, we utilize a pre-designed feature extractor to obtain the attributes feature $R^a \in \mathbb{R}^{|V| \times d_a}$ and directly combine it with the embedding feature R^e as a final vertex representation. So the dimension of vertex attributes must be greater than d_a to avoid introducing noise in the process of feature extraction. In this paper, we utilize the BOW (bag of words) model to obtain text vectors. According to the types of vertex attributes, users can flexibly design feature extractor. In our experiments, we take the hidden layer output of an autoencoder [16] as the attribute feature. The feature extractor and Transformer-decoder can be trained in parallel without affecting each other, which can greatly speed up the training process. The above process can be formulated as follows:

$$\begin{aligned} R^a &= feature\_extractor(A) \\ R &= [R^e|R^a] \end{aligned} \tag{4}$$

where $R \in \mathbb{R}^{|V|\times d}$ is the vertex representation matrix, $R^e \in \mathbb{R}^{|V|\times d_e}$ is the vertex embedding feature matrix, $R^a \in \mathbb{R}^{|V|\times d_a}$ is the attribute feature matrix, $[R^e|R^a]$ represents the operation of horizontally concatenating R^e and R^a. We assume that vertex attributes and network structure contribute equally to the vertex representation and set $d_a = d_e = \frac{d}{2}$, d is the dimension of vertex representation vector.

Attribute-Driven Laplacian Representation Optimization. In real-world networks, vertex attributes may have complex types, which do not apply to the Navie Combination described above. Some related works [1,22] use Laplacian Eigenmaps to enhance the ability of NRL models to retain network structure. Inspired by above works, we propose an Attributes-driven Laplacian Representation Optimization (Attr-LapRO) to make the proposed framework more general. We define the loss function of the Attr-LapRO as follows.

$$L_{lap} = \sum_{ij}(\mathbf{r_i} - \mathbf{r_j})^2 I_{ij} = 2 * Tr(R^T L R) \tag{5}$$

Where $R \in \mathbb{R}^{|V|\times d}$ is the vertex representation matrix, $I \in \mathbb{R}^{|V|\times|V|}$ is the vertex attribute similarity matrix, $I_{ij} \in [0,1]$ represents the attribute cosine similarity score of vertex v_i and v_j, $L = D - I$ is Laplacian eigenmap, $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix, $D_{ii} = \sum_j I_{ij}$. In this way, Attr-LapRO and Transformer-decoder share one output, so $R = R^e$, $d_e = d$.

We alternately and iteratively optimize the loss functions L_{lm} and L_{lap}, which is mainly for two reasons. One is that the parameters of the Transformer-decoder become challenging to update with two loss functions. The other reason is that the training of each stage can speed up the other's convergence process since vertices representation vectors are shared between the two stages.

Based on the two information fusion methods introduced above, we propose two types of TAFNE models: TAFNE$_{vanilla}$ and TAFNE$_{lap}$. TAFNE$_{vanilla}$ uses Naive Combination as the information fusion method, TAFNE$_{lap}$ utilizes Attr-LapRO to incorporate vertex attributes into the embedding process.

5 Evaluation

In this section, we first introduce the datasets used in this work and then validate the performance of our model compared to various state-of-the-art NRL algorithms through three downstream data mining tasks (i.e., clustering visualization, and node classification) on five datasets. Finally, we analyze the sensitivity of parameters.

5.1 Datasets

In order to fully evaluate the proposed method, we conduct experiments on three citation networks and two social networks with different sizes. Table 1 presents detailed information of the five datasets.

- Facebook [20] is a page-page graph of verified Facebook sites. Vertices represent official Facebook pages while the links are mutual likes between sites. Vertex attributes are extracted from the site descriptions that the page owners created to summarize the purpose of the site. All the vertices are divided into four categories, which are defined by Facebook.
- BlogCatalog [24] is a social network. Vertices represent bloggers, and categories of blogs written by bloggers are used as vertex labels. Each vertex has one or more labels. Edges represent friendship relationships between bloggers.
- Cora, CiteSeer, and PubMed [21] are citation networks, where each vertex represents a scientific publication, and the edge represents the citation relationship between vertices. Vertices are divided into different categories according to their research field, and each vertex has an abstract as its attribute.

5.2 Baseline Methods

We compare our method with several baseline methods, including DeepWalk [17], SDNE [26], Struc2Vec [19], DNE [22], TADW [28] and Attributes. In the Attributes method, the vertex attribute feature is treated as the vertex representation. Although there are other NRL methods, we can not list all of them. The methods mentioned above all have great innovations and conduct various verification experiments compared with other methods in the corresponding papers.

5.3 Parameter Settings

For all datasets, we set the dimension of the learned representation vector to $d = 128$. For the baseline methods, we follow the best parameter settings recommended in the original paper. In DeepWalk method, window size is $w = 10$, walk length is $l = 40$, and walks per vertex $\gamma = 40$. In Struc2Vec method, window size is $w = 10$, walk length is $l = 80$, walks per vertex is $\gamma = 10$. In SDNE method, the number of model layers is 3, and the hyperparameter $\alpha = 0.1$, $\beta = 10$. In DNE method, walk length is $l = 100$, walks per vertex is $\gamma = 100$, and the LSTM learning rate is 0.001. In TADW method, we set the parameters to the same as given in the corresponding paper. In Attributes method, the dimension of the vertex attribute feature is reduced to 128 via SVD [6]. In TAFNE method, we set the walk length to $l = 100$, walks per vertex $\gamma = 100$. Transformer-decoder has 8 layers and 4 masked self-attention heads per layer. At the stage of optimizing the loss function L_{lm}, we take advantage of Adam optimization scheme [8] with a max learning rate $lr_{lm_{max}} = 1e-4$. The learning rate was raised linearly from zero over the first 5000 updates and annealed to $1e-5$ using an exponential scheduler.

Table 1. Statistics of the datasets

Dataset	Nodes	Edges	Categories
Facebook	22470	171002	4
BlogCatalog	10312	333983	39
Cora	2707	5429	7
Citeseer	3311	4732	6
PubMed	19717	44338	3

Table 2. Clustering performance (NMI)

Methods	3-Cora	Cora
DeepWalk	0.045	0.012
SDNE	0.027	0.027
Struc2Vec	0.001	0.005
DNE	0.387	0.309
Attributes	0.451	0.286
TADW	0.613	0.411
TAFNE$_{vanilla}$	**0.655**	**0.468**
TAFNE$_{lap}$	0.648	0.447

5.4 Experiments Results

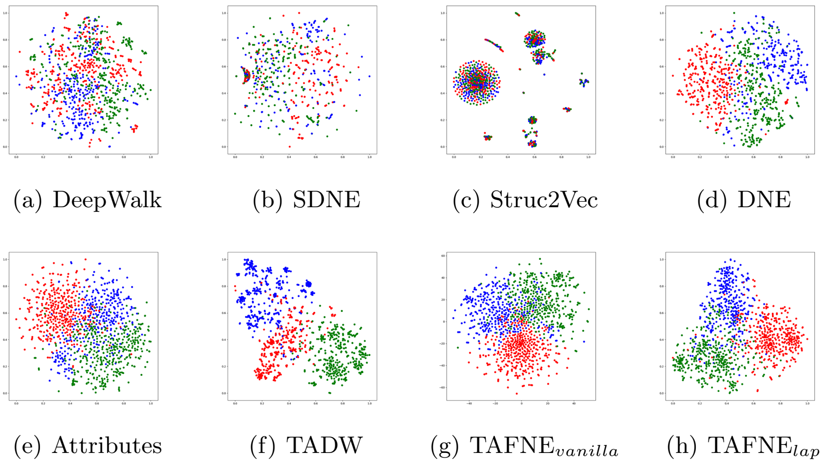
Clustering. In the real world, most data is unlabeled, and annotating data manually costs a lot, so learning the representation of a network is very important for unsupervised learning tasks. We perform clustering tasks on the 3-Cora and Cora datasets. 3-Cora is separated from Cora with 3 different categories.

In the clustering task, we use each baseline method to generate the vertices representation vectors that are used as features for clustering. The vertices are divided into several categories using the K-Means algorithm, and we evaluate the performance with NMI (Normalized Mutual Information)[3] score. Table 2 records the result of clustering. This result shows that TAFNE is significantly better than other methods. TADW achieves the best performance among baseline methods because of the consideration of both network topology and vertex attributes, but TAFNE still outperforms it. TAFNE$_{vanilla}$ is absolutely 0.042 and 0.057 better than TADW on the 3-Cora dataset and Cora dataset, respectively. Benefit from the combination with Transformer-decoder and information fusion model, TAFNE can retain network information more comprehensively. Therefore TAFNE is more robust in the clustering task.

Visualization. In the visualization task, we expect to reveal the network by visualizing the learned representations intuitively. We apply our method and baseline methods to the 3-Cora dataset, which has nearly 1300 vertices, and each vertex belongs to one of the three categories: Neural Network, Rule Learning, and Reinforcement Learning. We exploit t-SNE [11] to map the vertex representation learned by different methods to 2-dimension space. Figure 3 shows the visualization on the 3-Cora dataset, each point represents a scientific publication, and the different color represents a different category. The visualization of DeepWalk, SDNE, and Struc2Vec are not meaningful because the points of the same class are not clustered together. Although DNE and TADW can cluster most points of the same label together, the boundaries are not visible enough. The performance of TAFNE is much better than the baseline methods. Our method can not only cluster the points of the same category together, but also the clusters can be clearly separated from each other. This experiment indicates that TAFNE can learn more robust and informative representations.

Classification. In the node classification task, we perform multi-label and multi-class classification tasks on two types of datasets: one without vertex attributes (i.e., BlogCatalog) and one with vertex attributes (i.e., Facebook, Citeseer and PubMed). We treat the representations learned by various methods as the vertices feature vectors. For each dataset, a portion (L_V) of the labeled vertices are randomly sampled as the training data to train an MLP (Multilayer Perceptron) as the classifier, the rest of the vertices are the test data. We repeat this process 10 times and record the average performance in terms of classification accuracy and F1-score.

On the BlogCatalog dataset, we only focus on learning network topology feature to verify the effectiveness of the Transformer-decoder in capturing the


(a) DeepWalk (b) SDNE (c) Struc2Vec (d) DNE

(e) Attributes (f) TADW (g) TAFNE$_{vanilla}$ (h) TAFNE$_{lap}$

Fig. 3. Visualization of the 3-Cora dataset. Each point represents one scientific publication. Different colors correspond to different categories, i.e., Red: Nerual Network, Blue: Rule Learning, Green: Reinforcement Learning (Color figure online)

Table 3. Multi-label classification performance (Micro-F1/Macro-F1)

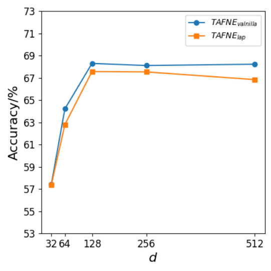
L_V	10%	20%	30%
DeepWalk	33.17/17.40	35.84/20.39	37.28/21.96
SDNE	31.25/11.54	32.27/14.33	32.24/16.31
Struc2Vec	11.16/5.24	11.20/5.57	12.86/4.83
DNE	34.13/17.38	37.25/21.41	37.56/23.06
TAFNE	**38.43/19.04**	**42.05/23.28**	**42.27/25.66**

Table 4. Multi-class classification performance (accuracy) on three datasets

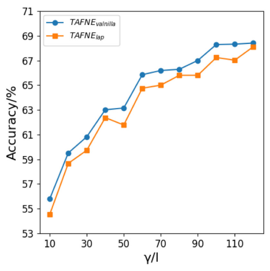
Dataset	Facebook			CiteSeer			PubMed		
L_V	10%	20%	30%	10%	20%	30%	10%	20%	30%
DeepWalk	72.09	74.83	76.14	50.94	51.54	53.28	69.94	71.37	72.51
SDNE	53.44	54.81	55.81	30.41	32.12	32.75	39.04	39.37	39.83
Struc2Vec	35.58	35.13	35.82	25.01	26.76	27.75	47.08	48.15	49.32
DNE	68.06	70.62	73.06	50.56	52.97	54.04	73.11	73.04	74.72
TADW	79.48	80.50	83.08	60.89	62.32	64.78	80.63	81.76	84.32
Attributes	74.27	76.62	78.41	57.31	59.20	61.09	75.45	77.52	78.79
TAFNE$_{vanilla}$	**82.23**	**82.96**	**85.28**	**63.20**	**65.75**	**67.01**	**83.19**	**84.67**	**86.94**
TAFNE$_{lap}$	80.14	81.45	83.23	62.18	64.13	65.61	81.83	84.27	85.23

network structure information without vertex attributes. Since TADW performs DeepWalk to preserve the network structure, so we exclude TADW. Then we perform a multi-label classification task on the vertices representations and report the performance with Micro-F1 and Macro-F1 scores of the classification results. From Table 3, one can see that TAFNE is 4.71% (Micro-F1) and 2.6% (Macro-F1) with 30% labeled vertices better than the best baseline DNE. The self-attention mechanism takes into account more contextual information, by which the model can better retain network global and local structure information.

Furthermore, we conduct multi-class classification tasks on the three attributed networks and evaluate the representations with classification accuracy. Table 4 documents the classification results. As we can see, TAFNE consistently outperforms other baseline methods. Compared to the best baseline TADW, TAFNE$_{vanilla}$ absolutely improves the accuracy by 2.20% on Facebook, 2.27% on CiteSeer, and 2.62% on PubMed. The classification results demonstrate the effectiveness of TAFNE to consider the network topology and vertex attribute information comprehensively, and the quality of the learned representation vectors are greatly improved.



(a) dimension



(b) walk per vertex/walk length

Fig. 4. Parameters sensitivity analysis of TAFNE on CiteSeer with train ratio as 50%

5.5 Paramter Sensitivity

In this section, we investigate the sensitivity of the method to the choice of parameter values. TAFNE has three main parameters: the dimension of the learned representation vector d, walk per vertex γ and walk length l. We test the classification accuracy with different parameters above CiteSeer dataset, and randomly sample 50% vertices as the trainset and the rest as the testset. We fix other parameters when inspecting each parameter.

Figure 4(a) shows the effect of different d on classification results. As the representation vector dimension d increases, the accuracy increases first and then decreases. The classification accuracy is highest when $d = 128$. For convenience we set $\gamma = l$. Figure 4(b) shows that with the increase of γ and l, the classification accuracy also improves, because γ and l correspond to the richness of the corpus. The bigger γ and l are, the richer the corpus is. Since the self-attention mechanism can solve the problem of long-distance dependency well, the increase in walk length not only does not have a negative impact but also better captures the global structure of the network.

6 Conclusion

To solve the problems of analyzing large-scale networks with computational overhead or intractability, and generate higher-quality vertices representations, this paper proposes an NRL framework TAFNE that can simultaneously incorporate network topology and vertex attribute information into the embedding process. The advantages of TAFNE can be summarized as follows:

(1) By utilizing the Transformer-decoder with masked self-attention mechanism, TAFNE can take more contextual information into account than previous models when extracting structural features from random walk vertex sequences. Thus TAFNE preserves global and local structural information better;

(2) For the networks where vertex contains textual attributes or numerical attributes of sufficient dimensions, TAFNE encodes the vertex attributes by an autoencoder and treats the attributes feature as part of the vertex representation. In this way, attributes feature directly measures the attribute-level similarity among vertices.
(3) For the networks where vertex contains attributes of complex types, TAFNE constructs an attribute similarity matrix to constrain the vertex representation. Such an approach makes TAFNE more generic.

Experimental results on the clustering, visualization, and node classification tasks over five datasets demonstrate the effectiveness of TAFNE.

References

1. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in Neural Information Processing Systems, pp. 585–591 (2002)
2. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput. **15**(6), 1373–1396 (2003)
3. Estévez, P.A., Tesmer, M., Perez, C.A., Zurada, J.M.: Normalized mutual information feature selection. IEEE Trans. Neural Netw. **20**(2), 189–201 (2009)
4. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 (2013)
5. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864 (2016)
6. Halko, N., Martinsson, P.G., Tropp, J.A.: Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. SIAM Rev. **53**(2), 217–288 (2011)
7. Henderson, K., et al.: RolX: structural role extraction & mining in large graphs. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1231–1239 (2012)
8. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
9. Krause, B., Kahembwe, E., Murray, I., Renals, S.: Dynamic evaluation of transformer language models. arXiv preprint arXiv:1904.08378 (2019)
10. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. J. Am. Soc. Inf. Sci. Technol. **58**(7), 1019–1031 (2007)
11. van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**(Nov), 2579–2605 (2008)
12. Malliaros, F.D., Vazirgiannis, M.: Clustering and community detection in directed networks: a survey. Phys. Rep. **533**(4), 95–142 (2013)
13. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
14. Natarajan, N., Dhillon, I.S.: Inductive matrix completion for predicting gene-disease associations. Bioinformatics **30**(12), i60–i68 (2014)
15. Newman, M.E.: Finding community structure in networks using the eigenvectors of matrices. Phys. Rev. E **74**(3), 036104 (2006)

16. Ng, A., et al.: Sparse autoencoder. CS294A Lecture notes **72**(2011), 1–19 (2011)
17. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710 (2014)
18. Rae, J.W., Potapenko, A., Jayakumar, S.M., Lillicrap, T.P.: Compressive transformers for long-range sequence modelling. arXiv preprint arXiv:1911.05507 (2019)
19. Ribeiro, L.F., Saverese, P.H., Figueiredo, D.R.: struc2vec: learning node representations from structural identity. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 385–394 (2017)
20. Rozemberczki, B., Allen, C., Sarkar, R.: Multi-scale attributed node embedding (2019)
21. Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., Eliassi-Rad, T.: Collective classification in network data. AI Mag. **29**(3), 93–93 (2008)
22. Sun, X., Song, Z., Dong, J., Yu, Y., Plant, C., Böhm, C.: Network structure and transfer behaviors embedding via deep prediction model. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 5041–5048 (2019)
23. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web, pp. 1067–1077 (2015)
24. Tang, L., Liu, H.: Relational learning via latent social dimensions. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 817–826 (2009)
25. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
26. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1225–1234 (2016)
27. Wang, X., He, X., Wang, M., Feng, F., Chua, T.S.: Neural graph collaborative filtering. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 165–174 (2019)
28. Yang, C., Liu, Z., Zhao, D., Sun, M., Chang, E.: Network representation learning with rich text information. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)
29. Zhang, D., Yin, J., Zhu, X., Zhang, C.: Network representation learning: a survey. IEEE Trans. Big Data **6**(1), 3–28 (2018)