A Simple Approach to Attributed Graph Embedding via Enhanced Autoencoder

Nasrullah Sheikh[1], Zekarias T. Kefato[2], and Alberto Montresor[1]

[1] University of Trento, Italy
{nasrullah.sheikh, alberto.montresor}@unitn.it
[2] Royal Institute of Technology, Stockholm, Sweden,
zekarias@kth.se

Abstract. Network Representation Learning (NRL) aims at learning a low-dimensional representation of nodes in a graph such that its properties are preserved in the learned embedding. NRL methods may exploit different sources of information such as the structural or attribute information of the graph. Recent efforts have shown that jointly using both structure and attributes helps in learning a better representation. Most of these methods rely on highly complex procedures, such as sampling, which makes them non-scalable to large graphs. In this paper, we propose a simple and scalable deep neural network model that learns an embedding by jointly incorporating the network structure and the attribute information. Specifically, the model employs an enhanced decoder that preserves global network structure and also handles the non-linearities of both the network structure and network attributes. We discuss node classification, link prediction, and network reconstruction experiments on four real-world datasets, demonstrating that our approach achieves better performance against the state-of-the-art baselines.

Keywords: Attributed Graphs, Network Embedding, Unsupervised Learning

1 Introduction

Graphs are ubiquitous. Artificial systems like the World Wide Web, Online Social Networks, power grids and communication networks, as well as natural phenomena such as protein-protein interaction networks and food chains: they all can be modeled as a collection of entities and their relationships.

Graphs play a critical role in various machine learning applications such as node classification [19], link prediction [12, 14] and recommendation [1, 3]. The performance of these applications depend on the capability of effectively extracting features from the input graphs. Computing traditional features such as graph statistics or local neighborhood structures is expensive and inflexible. Network Representation Learning [10,18,22,25,28] has emerged as an alternative approach to encode the information contained in the graph into a low-dimensional vector space such that the various relationships among nodes – such as high-order proximities – are preserved in the learned representation.

Early methods proposed for NRL were largely based on matrix factorization [2, 24]. Unfortunately, these models do not scale to large graphs due to the high computational complexity of factorization. Thanks to recent advances in the field of deep learning, neural NRL approaches have been proposed [10, 18, 25]. Most of these methods use only information about the network structure to learn a representation. Networks, however, are often provided with other useful information, such as attributes associated with each of the nodes.

Recent works have shown that exploiting this additional information improves the performance of the aforementioned machine learning tasks [4, 10, 20, 26, 28]. This is because nodes tend to have homophilic relationships, i.e. they tend to connect to nodes that are similar to themselves in terms of attributes. By learning from both the network structure and the attributes at the same time, better representations can be obtained. But this comes at a cost, due to the increasing computation complexity needed to model the various relationships, structural and attribute, between nodes.

For example, recent papers such as Deep Attributed Network Embedding (DANE) [4] employ a deep autoencoder model of the network structure and attributes. This method preserves semantic proximities between nodes by sampling nodes that have high similarity in attributes and optimizing their representation together with the first- and higher-order proximities. Unfortunately, the cost of this sampling procedure is quadratic in the number of nodes, thus not scalable to large graphs. Moreover, for some graphs the number of attributes may be very large (in order of millions), that will increase the overall model complexity. Another approach called Attributed Network Representation Learning (ANRL) [28] uses a neighbor enhancement autoencoder which optimizes the learning on attributes and the neighborhood context. The complexity of the generating neighborhood context is too high for large graphs, both in terms of time and space.

It is important to note that, similar to the network structure, attributes are also highly-nonlinear and sparse [13, 25]. DANE and ANRL handle such nonlinearity by employing a deep model on attributes, at the expense of high sampling complexity, as mentioned above. Handling attributes in conjunction with structure is even more challenging as it involves capturing non-linearity and sparsity, while at the same time preserving the structural properties of the graph.

To address these problems, we propose SAGE2VEC (Simple Attributed Graph Embedding), a model that preserves the second-order proximities in the structure, handles the structure and attribute non-linearity and sparsity, and has linear complexity w.r.t to the number of nodes. The motivation behind this work is to build a simple model in terms of number of parameters and to show that a very simple approach can achieve better results than complex state-of-the-art baselines in a large percentage of scenarios. We employ an autoencoder model with an *enhanced decoder* that takes only structural information as input but optimizes on both structure and attribute information. By using attribute information for optimization reduces the overall model complexity and thus, the model can be scaled to larger graphs.

2 Related Work

NRL methods are classified in two groups: plain methods that use the structural information only and attributed-based methods that use attributes as well.

Plain NRL Methods Various approaches have suggested to use the topological information of networks to learn their representation [5, 18, 23, 25]. Earlier methods focused on matrix-factorization approaches [2, 23]. These methods involve eigen decomposition to generate an embedding; e.g., Tang et al. use the top-d eigenvectors of the modularity matrix as embedding [23]. These methods are not scalable to large graphs due the computational costs of factorization.

Alternative approaches have taken inspiration from Natural Language Processing (NLP), opportunely adapted to graph embeddings. Perozzi et al. proposed DEEPWALK, an unbiased random walk based approach to learn an embedding based on SKIPGRAM [16] that preserves the second-order proximity [18]. Another approach called NODE2VEC is based on biased random walks and uses two hyper-parameters p and q to interpolate between breadth-first search and depth-first search [5]. LINE learns a representation that preserves first- and second-order proximity [22]. Wang et al. proposed SDNE that captures the high non-linearity of the network structure by using a deep autoencoder capable to capture both first- and second-order proximities [25].

Attributed NRL Methods Graphs are often enriched with additional information such as attributes which, if exploited, enable to learn better representations. Yang et al. proposed TADW, a matrix factorization method that uses both the network structure and the text features of nodes [26]. It is computationally expensive as it applies SVD on the text matrix. The graph Laplacian-based approach proposed by Huang et al. uses structure and text matrix to learn an unified embedding in a distributed manner by dividing the complex optimizations into many sub-problems [7]. Sheikh et al. proposed GAT2VEC, that creates a bipartite graph on attributes and then jointly learns a representation from network structure and bipartite attributed network [20]. Some approaches use partial labels–if available–for learning [9,17,21,27]. TRIDNR learns two separated embeddings on network structure and attributed using DEEPWALK and DOC2VEC [11]. A semi-supervised method by Kipf and Welling uses graph convolutional neural networks (GCN) [9]. Variational Graph Auto-encoder (VGAE) is a unsupervised architecture that uses the GCN encoder and an inner product decoder to learn a representation using network structure and attributes [10].

Various deep neural network models have been proposed, based on autoencoders that capture the non-linearity of structure/attributes and preserve the various proximities [4, 15, 28]. DANE uses a deep autoencoder on structure and attributes; its embeddings are optimized based on various proximities [4]. ANRL uses a neighbor enhancement decoder model that takes node features as input, while reconstructing the target neighbors [28]. Meng et al. proposed CAN, that learns an embedding of both the nodes and the attributes in the same space using a variational graph autoencoder such that the affinities between the network structure and the attributes are properly preserved and measured [15].

3 Preliminaries and Problem Definition

Before proceeding to the definition of the problem, we introduce the notation used throughout the paper. Matrices and vectors are denoted by uppercase and lowercase boldface letters, respectively; a boldface letter with a single subscript denotes a row vector of the respective matrix, while a boldface letter with two subscripts represents the scalar element of the matrix.

Let $G = (V, \mathbf{A}, \mathbf{Z})$ be an *attributed graph* where V is a set containing N vertices, $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix describing the edges and $\mathbf{Z} \in \mathbb{R}^{N \times F}$ is the matrix representing F features potentially associated to the N nodes.

Let $\mathbf{A}_i$ be the i^{th} row vector of $\mathbf{A}$ and let $\mathbf{A}_{ij} > 0$ if there is an edge between vertexes i and j. Let $\mathbf{Z}_i$ be the feature vector of vertex i, where $\mathbf{Z}_{iu} > 0$ specifies whether vertex i is associated to attribute u.

Problem Given an attributed graph $G = (V, \mathbf{A}, \mathbf{Z})$, we aim at learning a low-dimensional network representation $\Phi : V \to \mathbb{R}^d$, where $d \ll |V|$ is the dimension of the learned representation, such that the mapping function Φ preserves the structural and attribute information.

The quality of this mapping will be evaluated against several tasks such as node classification on various real-world datasets, as described in Section 5.

4 Model

We describe now Sage2Vec, a novel model that learns an embedding using both the network structure and the attributes. It is essential that the learning occurs on both elements at the same time–and not as a combination of two learned models–so that the two modalities complement each other. In this way, the learned representation preserves the structural proximity and encodes the attribute information in it.

We propose an autoencoder with an enhanced decoder model that jointly learns a representation using the network structure and the attributes. Our model preserves the global network structure, captures the non-linearity and addresses the sparsity of both the network structure and the node attributes.

For the sake of completeness, we briefly describe the autoencoder model. It is an unsupervised model consisting of two parts: encoder and decoder. Both of them consist of multiple non-linear functions which map the input data to the representation space (encoder) and from the representation space to the reconstruction space (decoder). Since, the degree distribution of $\mathbf{A}$ varies, we use the normalization trick introduced by Kipf et al. [9] as: $\mathbf{M} = \hat{\mathbf{D}}^{-\frac{1}{2}} \hat{\mathbf{A}} \hat{\mathbf{D}}^{-\frac{1}{2}}$, where $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\mathbf{I}$ is the identity matrix and $\hat{\mathbf{D}}$ is the diagonal degree matrix of $\hat{\mathbf{A}}$. Specifically, given a vector $\mathbf{M}_i$ of the i^{th} vertex as input, the encoder is described as follows:

$$\begin{aligned} \mathbf{Y}_i^{(1)} &= \sigma(\mathbf{W}^{(1)} \mathbf{M}_i + b^{(1)}) \\ \mathbf{Y}_i^{(l)} &= \sigma(\mathbf{W}^{(l)} \mathbf{Y}_i^{(l-1)} + b^{(l)}), \quad l = 2, \cdots, L \end{aligned} \tag{1}$$

$\mathbf{Y}_i^l$ is the latent representation of $\mathbf{M}_i$ at the l^{th} layer; $\sigma(.)$ is a non-linear function such as $tanh, relu$; L is the number of layers; $\mathbf{W}^{(l)}$ is the weight matrix and $b^{(l)}$ is the bias at the l^{th} layer. The decoder is the reverse of the encoder; thus, it takes $\mathbf{Y}_i^{(l)}$ as input and produces the reconstruction $\tilde{\mathbf{M}}_i$. The objective of the autoencoder is to minimize the reconstruction error, defined as:

$$\mathcal{L} = \textstyle\sum_{i=1}^{N} ||\mathbf{M}_i - \tilde{\mathbf{M}}_i||_2^2 \tag{2}$$

Enhanced Autoencoder Our enhanced autoencoder model considers the non-linearity of network attributes in conjunction with the network structure and jointly learns a representation from these two different modalities. Its architecture is shown in Fig. 1. The encoder part is similar to Equation 1. Our model only input is the network information. For a given vertex i, our model takes its local neighborhood information $\mathbf{M}_i$ as input, and aims to reconstruct its neighbors $\tilde{\mathbf{M}}_i$ along with its respective attributes $\tilde{\mathbf{Z}}_i$ by incorporating prior knowledge of attributes i.e, $\mathbf{Z}_i$. To accommodate the reconstruction of attributes, Equation 2 is modified as follows:

$$\mathcal{L}_{ea} = \sum_{i=1}^{N} \|(\mathbf{M}_i - \tilde{\mathbf{M}}_i)\|_2^2 + \sum_{i=1}^{N} \|(\mathbf{Z}_i - \tilde{\mathbf{Z}}_i)\|_2^2 \tag{3}$$

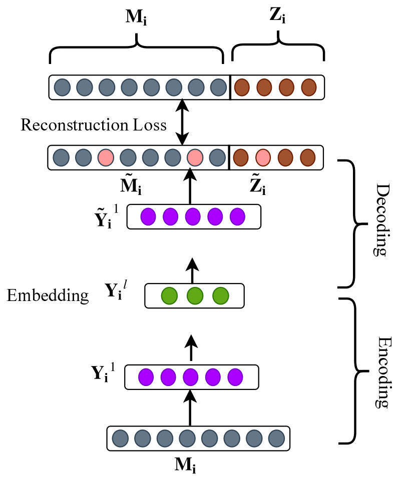

Fig. 1. The architecture of our proposed model

Since both the network and the attributes are sparse, the autoencoder tends to reconstruct zeros. Similar to Wang et al., we impose a higher penalty to the reconstruction error of the non-zero elements than zero elements [25]. To incorporate the penalties, Equation 3 is revised as:

$$\begin{aligned}\mathcal{L}_{ea} &= \sum_{i=1}^{N} \|(\mathbf{M}_i - \tilde{\mathbf{M}}_i) \odot \mathbf{C}_i^s\|_2^2 + \sum_{i=1}^{N} \|(\mathbf{Z}_i - \tilde{\mathbf{Z}}_i) \odot \mathbf{C}_i^a\|_2^2 \\ &= \|(\mathbf{M} - \tilde{\mathbf{M}}) \odot \mathbf{C}^s\|_F^2 + \|(\mathbf{Z} - \tilde{\mathbf{Z}}) \odot \mathbf{C}^a\|_F^2\end{aligned} \tag{4}$$

where $\odot$ is the Hadamard product, $\mathbf{C}_i^s = [\mathbf{C}_{i1}^s, \cdots, \mathbf{C}_{iN}^s]$ and $\mathbf{C}_i^a = [\mathbf{C}_{i1}^a, \cdots, \mathbf{C}_{iF}^a]$ are the penalties for network and attribute zero reconstructions, respectively. The penalty for zero reconstruction when the dealing with the network structure is given as:

$$C_{ij}^s = \begin{cases} 1, & \text{if } \mathbf{M}_{ij} = 0 \\ \beta, & \text{otherwise} \end{cases} \tag{5}$$

where $\beta > 1$. Likewise, the penalties for zero reconstructions in attributes are similar to Equation 5.

To prevent overfitting, we add an $l2$ regularizer. Thus, the overall objective is to minimize the following loss function:

$$\begin{aligned}\mathcal{L}_o &= \mathcal{L}_{ea} + \gamma\mathcal{L}_{reg} \\ &= \|(\mathbf{M} - \tilde{\mathbf{M}}) \odot \mathbf{C}^s\|_F^2 + \|(\mathbf{Z} - \tilde{\mathbf{Z}}) \odot \mathbf{C}^a\|_F^2 + \gamma \sum_{l=1}^{L}(\|\mathbf{W}^{(l)}\|_F^2 + \|\tilde{\mathbf{W}}^{(l)}\|_F^2)\end{aligned} \quad (6)$$

where $\mathbf{W}^{(l)}$ and $\tilde{\mathbf{W}}^{(l)}$ are the weight matrices at l-th layer encoder and decoder, respectively, while γ is the regularization coefficient.

We use stochastic gradient descent to minimize the objective function $\mathcal{L}_o$ by using back-propagation on the model parameters $\theta = \{\mathbf{W}^{(i)}, \tilde{\mathbf{W}}^{(i)}, \mathbf{B}^{(i)}\}$.

5 Experiments

In this section, we describe the datasets and machine learning tasks used to evaluate our proposed approach against the state-of-the-art baselines.

5.1 Datasets

We conducted our experiments on four benchmark datasets (Table 1). CITESEER, CORA and PUBMED[3] are citation networks. CORA contains papers about machine learning, grouped in seven categories. CITESEER contains papers from six categories corresponding to computer science fields. In both CITESEER and CORA, the attributes are 0/1 valued vectors for each node. The PUBMED dataset is a citation network related to diabetes. WIKI is a web graph of hyperlinks [4]. The attribute vector of PUBMED and WIKI is a TF/IDF word vector.

Table 1. Dataset statistics

Datasets	$\|V\|$	$\|E\|$	F	#Labels
CITESEER	3,312	4,660	3,703	6
CORA	2,708	5,278	1,433	7
PUBMED	19,717	44,338	500	3
WIKI	2,405	12,761	4,973	17

5.2 Baselines

We evaluate SAGE2VEC with several state-of-the-art baseline methods. DEEPWALK and NODE2VEC use only the structure of the network, while the rest use both the structure and the attributes to learn an embedding.

- DEEPWALK (Dw) [18] employs short random walks to generate a corpus of vertex sequences, and then uses SKIPGRAM to learn a representation.
- NODE2VEC (N2v) [5] uses biased short random walks to explore the diverse neighborhood to interpolate between *breadth-first* and *depth-first* sampling.
- VGAE [10] is based on *variational graph encoders* using graph convolutional network (GCN) and exploit both structure and attributes.
- GAE [10] is an non-probabilistic variant of VGAE.

[3] https://linqs.soe.ucsc.edu/data

– DANE [4] uses a deep network on both structural and attribute information, while maintaining the consistent and complimentary information between the two modalities of informations.
– ANRL [28] uses an attribute-aware SKIPGRAM model to incorporate attribute information and a neighbor-enhanced autoencoder to reconstruct the target neighbors. The best-performing variant ANRL-WAN of ANRL is used here.
– CAN [15] uses a variational autoencoder to learn an embedding of vertices and attributes in the same semantic space. For evaluation, we use only the learned representations of nodes.

Table 2. The Network Layer Structure for Enhanced Autoencoder

Dataset	#neurons in each layer
CITESEER	3312 - 128 - 7015
CORA	2708 - 128 - 4141
PUBMED	19717 - 128 - 19217
WIKI	2405 - 128 - 7378

We used the code of the baselines released by the authors. The parameters used are the reported optimal parameters, otherwise we performed a random search to obtain an optimal performance of the baseline for a given dataset. For our model, the number of layers and their sizes for four datasets is given in Table 2, the activation function is *tanh*, the *Adam* Optimizer [8] is employed, and the embedding size d is 128. Our model requires very less trainable parameters as can be seen in Table 2, thus can be trained quickly.

5.3 Vertex Classification

We evaluate the learned representations of our approach against baselines through multi-class classification. We use *one-vs-rest* l_2 regularized logistic regression classifier. We randomly selected $T_r \in \{10\%, 30\%, 50\%\}$ vertices as training set and the rest as test set. We repeated the process 10 times for each T_r, and report the average MICRO-F1 (Mi-F1) and MACRO-F1 (Ma-F1).

The classification results of all datasets are shown in Table 3. The best values are highlighted in bold. SAGE2VEC achieves the best performance against all baselines in the CORA, PUBMED, WIKI datasets and has competitive performance with ANRL for the CITESEER dataset. Specifically, our model achieves an improvement around 11% in MACRO-F1 for the WIKI dataset against the second-best performing baseline (DANE). The results show that a well-designed model leveraging both structure and attributes can learn a better representation without requiring complex and computationally-intensive optimizations.

5.4 Link Prediction

Following [5, 10], we generate a residual network by removing 10% of edges from the network, retaining all features and ensuring that the network remains connected. These removed edges are called *true edges.* We sample an equal percentage of missing edges (non-edges), called *false edges.* True edges and false edges form a test set. We train the NRL models on the residual network.

Table 3. Micro-F1 and Macro-F1 scores for Vertex Classification

Tr. Ratio	Metric	Dw	N2v	gae	vgae	dane	anrl	can	Sage2Vec
		citeseer							
10%	Mi-F1	49.48	53.92	57.84	59.43	66.22	**72.24**	68.30	71.81
	Ma-F1	42.84	46.48	50.34	52.40	56.60	**63.60**	59.82	62.75
30%	Mi-F1	53.85	56.07	58.50	61.55	69.92	73.46	70.63	**73.74**
	Ma-F1	46.64	48.60	50.88	54.67	61.48	67.15	62.30	**67.30**
50%	Mi-F1	55.36	57.37	58.75	61.87	71.92	73.28	71.50	**74.89**
	Ma-F1	48.60	50.00	51.50	55.19	64.87	68.32	63.44	**69.42**
		cora							
10%	Mi-F1	70.89	74.00	73.54	76.45	77.80	73.55	78.68	**79.97**
	Ma-F1	67.55	70.66	70.00	74.06	73.75	69.51	75.55	**77.05**
30%	Mi-F1	76.30	78.15	76.39	78.54	81.97	76.55	82.22	**84.45**
	Ma-F1	74.62	77.17	72.75	77.72	80.45	73.85	79.48	**83.15**
50%	Mi-F1	78.00	79.32	76.54	78.84	82.97	78.24	82.57	**85.67**
	Ma-F1	76.60	78.60	73.31	77.45	81.42	74.50	80.77	**84.0**
		pubmed							
10%	Mi-F1	74.65	80.27	81.53	80.21	84.90	84.55	78.52	**85.60**
	Ma-F1	72.90	78.77	80.97	79.52	84.22	84.69	78.20	**85.28**
30%	Mi-F1	75.72	80.90	81.84	80.50	85.27	85.11	79.05	**86.21**
	Ma-F1	74.28	79.52	81.32	79.83	85.24	85.25	78.65	**85.89**
50%	Mi-F1	76.13	81.20	82.01	80.70	**86.76**	86.32	79.38	86.55
	Ma-F1	74.76	79.85	81.50	80.05	86.42	**86.46**	78.97	86.22
		wiki							
10%	Mi-F1	54.78	54.22	57.04	60.77	65.13	59.15	56.73	**68.79**
	Ma-F1	35.86	34.08	39.22	40.51	44.25	39.41	36.97	**48.15**
30%	Mi-F1	61.25	61.06	64.74	64.13	73.85	67.80	64.89	**77.01**
	Ma-F1	41.80	40.54	46.88	45.04	54.0	49.03	44.99	**57.80**
50%	Mi-F1	62.54	62.47	67.11	65.16	75.95	70.05	67.48	**79.42**
	Ma-F1	44.25	42.50	49.97	46.47	56.82	52.98	48.85	**63.15**

The models are evaluated based on their ability to correctly classify true edges and false edges. Similar to [28], we classify the true and false edges according to a ranking based on cosine similarity. We employ *area under ROC curve* (AUC) and *average precision* (AP) to evaluate the NRL models on the test set. We perform link prediction on all four datasets, showing the results in Table 4. Once again, the results show the advantages of using structure and attribute information jointly. These two modalities complement each other and the network can show structure homophily, attribute homophily or both. Our model performs better in AUC and AP for the citeseer, pubmed datasets and is comparable for cora and wiki against all the baselines. In particular, our method achieves 2% gains in AUC and AP against the state-of-the-art baseline for the pubmed dataset.

Table 4. AUC & AP scores for Link Prediction

Method	CITESEER		CORA		PUBMED		WIKI	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP
DEEPWALK	0.83	0.87	0.86	0.89	0.84	0.86	0.90	0.92
NODE2VEC	0.85	0.87	0.89	0.91	0.93	0.93	0.87	0.90
GAE	0.90	0.89	**0.92**	0.92	0.94	0.94	0.91	0.93
VGAE	0.93	0.94	0.92	**0.93**	0.92	0.92	0.92	0.94
DANE	0.85	0.88	0.78	0.81	0.84	0.86	0.86	0.87
ANRL	0.95	0.94	0.87	0.86	0.92	0.92	0.94	0.94
CAN	0.96	0.96	**0.92**	**0.93**	0.94	0.93	**0.96**	**0.97**
SAGE2VEC	**0.97**	**0.97**	**0.92**	**0.93**	**0.96**	**0.96**	0.95	0.96

5.5 Network Reconstruction

The purpose of network reconstruction is to test the preservation of the network structure in the learned embeddings. For each dataset, we learned an embedding using the baseline methods to predict the edges. Then we compute the probability of an edge $p(i, j)$ for each pair of node i, j given as:

$$p(i, j) = \frac{1}{1 + exp^{-(\Phi(i).\Phi(j)^T)}} \tag{7}$$

where T is the transpose operation. We rank these edges from high to low probabilities. The edges from the original network serve as ground truth. We evaluate the performance of network reconstruction by correctly predicting the edges using *precision-at-K* ($P@K$) metric against the ground truth. K represents the number of predicted edges selected for evaluation. The results of Table 5 show that our method performs better over the CITESEER, CORA and WIKI datasets. In all these datasets, when K increases, $P@K$ is consistently higher than all baselines. In case of PUBMED, since the dataset is sparse, the NODE2VEC is able to explore properly the diverse neighborhood.

5.6 Algorithmic and Scalability Analysis

In this section, we provide an analysis of the computational complexity of the baselines and our approach. The algorithmic computational complexity of the state-of-the-art models is provided in Table 6. In case of DANE, authors have used some simple sampling strategy but the algorithm is still dominated by a quadratic time complexity. Given that real-world large graphs have nodes and edges in the order of millions, the state-of-the-art methods require huge computational resources, thus limiting their usability. Our model SAGE2VEC has the smallest algorithmic computational complexity.

To perform the scalability analysis we used the larger graph dataset available, REDDIT[4]. It has 232K nodes, 11M edges and 602 attribute vector. The REDDIT is constructed from the reddit posts where a node is a post, and two posts

[4] http://snap.stanford.edu/graphsage/#datasets

Table 5. Network Reconstruction Precision at K (P@K)

Dataset	K	Dw	N2v	gae	vgae	dane	anrl	can	Sage2Vec
citeseer	100	0.11	0.3	0.44	0.51	0.04	0.17	0.57	**0.58**
	1000	0.14	0.5	0.3	0.39	0.03	0.08	0.46	**0.58**
	3000	0.13	0.34	0.25	0.3	0.04	0.07	0.39	**0.53**
	5000	0.14	0.31	0.21	0.26	0.19	0.06	0.37	**0.49**
	10000	0.14	0.23	0.16	0.22	0.22	0.05	0.33	**0.42**
cora	100	0.1	0.52	0.49	**0.54**	0.0	0.08	0.52	0.43
	1000	0.16	0.36	0.47	0.46	0.0	0.1	0.51	**0.55**
	3000	0.15	0.18	0.37	0.39	0.05	0.07	0.45	**0.52**
	5000	0.15	0.14	0.33	0.35	0.17	0.06	0.42	**0.47**
	10000	0.14	0.12	0.27	0.28	0.19	0.05	**0.36**	**0.36**
pubmed	100	0.01	**0.36**	0.31	0.28	0.0	0.04	0.32	0.26
	1000	0.03	**0.51**	0.24	0.18	0.0	0.03	0.18	0.14
	3000	0.03	**0.43**	0.2	0.14	0.0	0.03	0.15	0.14
	5000	0.03	**0.40**	0.18	0.13	0.0	0.03	0.14	0.13
	10000	0.03	**0.33**	0.16	0.11	0.0	0.02	0.12	0.12
wiki	100	0.43	0.23	0.06	**0.92**	0.5	0.08	0.51	0.87
	1000	0.33	0.04	0.04	**0.88**	0.47	0.16	0.23	**0.88**
	3000	0.32	0.09	0.04	0.78	0.55	0.19	0.2	**0.83**
	5000	0.3	0.1	0.04	0.72	0.66	0.18	0.19	**0.77**
	10000	0.28	0.1	0.05	0.58	0.58	0.19	0.15	**0.66**

are connected if the same user comments on both posts. The attributes are a concatenation of average word vectors of post's title and post's comment; post's score and the number of comments on the post [6].

We conducted the experiments on a 48-core machine based on the Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz processor, with 128 GB of RAM. The experiment was conducted on CPU due to constraints in GPU memory size. We report the asymptotic computational complexity of the algorithms and the wall-clock time in the Table 6. Though anrl has lowest complexity, but as mentioned previously it has high preprocessing cost for neighborhood context generation. Though, gae has linear a linear time complexity, but it does not scale because in large graphs the number of edges are far greater than the number of nodes. For running time analysis, all baseline methods ran out of memory; dane and anrl during the data preprocessing phase, while gae and can during the model computation. We run Sage2Vec[5] for 5 epochs to show that the model uses less parameters and thus can be used for large graphs.

6 Conclusion

In this work, we proposed an enhanced autoencoder model that jointly exploits both the network structure and the attribute information, with the aim of learning an embedding of the graph. We designed an enhanced decoder which pre-

[5] neuron in layers: 232,965 - 1000 - 64 - 1000 - 233,567

Table 6. Computational Complexity and Scalability Analysis on REDDIT dataset (NA:*out of memory*)

Dataset	**Algorithmic Computational Complexity**	**Training Time**
GAE	$O(\|E\|)$	NA
DANE	$O(N^2)$	NA
ANRL	$O(N \log N)$	NA
CAN	$O(\|E\|)$	NA
SAGE2VEC	$O(N)$	≈10 hours

serves global network structure and at the same time handles the non-linearity and sparsity of both network structure and attributes. Furthermore, our model requires a smaller number of trainable parameters as compared to the state-of-the-art baselines and therefore can be easily trained for large datasets. The quality of learned representation is verified through extensive experiments on four real-world datasets and our proposed approach outperforms the state-of-the-art baselines in most of the cases. Thus, our approach shows that a well-designed but simple model can achieve a very good performance.

References

1. L. Backstrom and J. Leskovec. Supervised random walks: predicting and recommending links in social networks. In *Proc. of the 4th ACM Int. Conf. on Web search and data mining*, pages 635–644. ACM, 2011.
2. M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proc. of the 14th Int. Conf. on Neural Information Processing Systems*, NIPS'01, pages 585–591. MIT Press, 2001.
3. F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans. on Knowl. and Data Eng.*, 19(3):355–369, 2007.
4. H. Gao and H. Huang. Deep attributed network embedding. In *Proc. of the 27th Int. Conf. on Artificial Intelligence, IJCAI-18*, pages 3364–3370, 7 2018.
5. A. Grover and J. Leskovec. Node2vec: Scalable feature learning for networks. In *Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, KDD '16, pages 855–864. ACM, 2016.
6. W. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017.
7. X. Huang, J. Li, and X. Hu. Accelerated attributed network embedding. In *SIAM Int. Conf. on Data Mining*, pages 633–641, 2017.
8. D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.
9. T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.
10. T. N. Kipf and M. Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

11. Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. *CoRR*, 2014.
12. D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.*, 58(7):1019–1031, 2007.
13. D. Luo, C. H. Q. Ding, F. Nie, and H. Huang. Cauchy graph embedding. In *Proc. of the 28th Int. Conf. on Machine Learning*, ICML'11, pages 553–560, 2011.
14. L. Lü and T. Zhou. Link prediction in complex networks: A survey. *Physica A*, 390(6):11501170, 2011.
15. Z. Meng, S. Liang, H. Bao, and X. Zhang. Co-embedding attributed networks. In *Proc. of the 12th ACM Int. Conf. on Web Search and Data Mining*, WSDM '19. ACM, 2019.
16. T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.
17. S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang. Tri-party deep network representation. In *Proc. of the 25th Int. Conf. on Artificial Intelligence*, IJCAI'16, pages 1895–1901. AAAI Press, 2016.
18. B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proc. of the 20th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710. ACM, 2014.
19. P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.
20. N. Sheikh, Z. Kefato, and A. Montresor. GAT2VEC: Representation learning for attributed graphs. *Computing*, Apr 2018.
21. J. Tang, M. Qu, and Q. Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proc. of the 21th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, KDD'15, pages 1165–1174. ACM, 2015.
22. J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. LINE: Large-scale information network embedding. In *Proc. of the 24th Int. Conf. on World Wide Web*, WWW '15, pages 1067–1077, 2015.
23. L. Tang and H. Liu. Relational learning via latent social dimensions. In *Proc. of the 15th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, KDD '09, pages 817–826. ACM, 2009.
24. J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319, 2000.
25. D. Wang, P. Cui, and W. Zhu. Structural deep network embedding. In *Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, KDD '16, pages 1225–1234. ACM, 2016.
26. C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Chang. Network representation learning with rich text information. In *Proc. of the 24th Int. Conf. on Artificial Intelligence*, IJCAI'15, pages 2111–2117. AAAI Press, 2015.
27. Z. Yang, W. Cohen, and W. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *CoRR*, abs/1603.08861, 2016.
28. Z. Zhang, H. Yang, J. Bu, S. Zhou, P. Yu, J. Zhang, M. Ester, and C. Wang. Anrl: Attributed network representation learning via deep neural networks. In *Proc. of the 27th Int. Conf. on Artificial Intelligence, IJCAI-18*, pages 3155–3161, 7 2018.