CrossMark

GAT2VEC: representation learning for attributed graphs

Nasrullah Sheikh[1] · **Zekarias Kefato**[1] ·
Alberto Montresor[1]

Abstract Network representation learning (NRL) enables the application of machine learning tasks such as classification, prediction and recommendation to networks. Apart from their graph structure, networks are often associated with diverse information in the form of attributes. Most NRL methods have focused just on structural information, and separately apply a traditional representation learning on attributes. When multiple sources of information are available, using a combination of them may be beneficial as they complement each other in generating accurate contexts; moreover, their combined use may be fundamental when the information sources are sparse. The learning methods should thus preserve both the structural and attribute aspects. In this paper, we investigate how attributes can be modeled, and subsequently used along with structural information in learning the representation. We introduce the GAT2VEC framework that uses structural information to generate structural contexts, attributes to generate attribute contexts, and employs a shallow neural network model to learn a joint representation from them. We evaluate our proposed method against state-of-the-art baselines, using real-world datasets on vertex classification (multi-class and multi-label), link-prediction, and visualization tasks. The experiments show that GAT2VEC is effective in exploiting multiple sources of information, thus learning accurate representations and outperforming the state-of-the-art in the aforementioned

✉ Nasrullah Sheikh
  nasrullah.sheikh@unitn.it

  Zekarias Kefato
  zekarias.kefato@unitn.it

  Alberto Montresor
  alberto.montresor@unitn.it

[1] University of Trento, Trento, Italy

tasks. Finally, we perform query tasks on learned representation and show how the qualitative analysis of results has better performance as well.

Keywords Attributed graphs · Network embedding · Unsupervised learning · Deep learning

Mathematics Subject Classification 68T99 · 82C32 · 82C41

1 Introduction

The structural relationships between entities in domains as diverse as the world wide web, online social networks and computer networks can be modeled as graphs. The entities are represented as nodes and the interactions between them are represented as edges. The sparsity of these large real-world graphs poses serious challenges to machine learning tasks such as vertex classification [20], recommendation [8,31], anomaly detection [3], visualization [22], and link prediction [15]. Network representation learning (NRL) [9,19,24] has been recently proposed to address the sparsity problem by learning continuous, low-dimensional latent features of vertices. These features capture the global structure of a network and can be later fetched to machine learning tasks.

An emerging approach to NRL is based on applying models developed for natural language processing to networks. A famous quote by J.R. Firth says, "You shall know a word by the company it keeps". Following this motto, the SKIPGRAM [16] algorithm learns embeddings for textual data (e.g., sentences) and contextual relationship between words is preserved in the learned embedding.

Recently, the DEEPWALK algorithm has been the first to apply the language model to networks. Sequences of nodes are obtained by short random walks [19], which are then fed to SKIPGRAM. Intuitively, close nodes will tend to have similar contexts (sequences) and thus have embeddings that are close to each other. This basic idea has been later expanded in several ways [9,24].

NRL methods have largely focused only on the structure of the graphs and have paid less attention to other potential sources of information. Modern networks may have rich content associated with nodes or edges, usually in the form of attributes. For example, in a social network where nodes are people and edges are (follower/friendship) relationships between them, attributes may include the group memberships, the school attended, the workplace, etc.

Attributes are invaluable in cases where structural information is missing, or when structurally unrelated vertices have high attribute similarity. For example, in Fig. 1, vertex 9 is disconnected, but with the aid of attributes, it can have a representation similar to vertex 6. In the same way, vertices 1 and 8 are structurally far away from each other, but they have similar attributes, thus they should be close to each other in a low-dimensional space.

Similarly to structural contexts, we can define the context of vertices in terms of attributes, but it is a challenging task to generate attribute contexts from attributed graphs, in particular when the coverage of attributes is only partial, as in Fig. 1. The labels of vertices are used in the classification task.
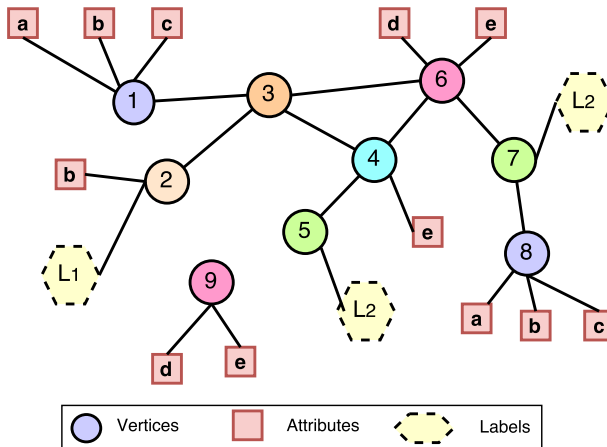
Fig. 1 An example of a partially attributed graph

In this paper, we introduce the GAT2VEC framework that jointly learns from both the structure and the attributes of the network using a single neural layer. Structural contexts are obtained from the graph, preserving structural proximities; attribute contexts are obtained from a bipartite graph linking vertices and their attributes, preserving content proximities.

The proposed framework learns a representation in an unsupervised manner, scaling to large graphs, for both directed and undirected homogeneous graphs. Our approach is novel as it leverages multiple sources of information through early fusion and needs to optimize a single objective function. Furthermore, we will present a semi-supervised variant of GAT2VEC called GAT2VEC- WL, where labels are incorporated as attributes to enhance the learning of embeddings.

We empirically evaluated and validated our approach on vertex classification (multi-class and multi-label) and link prediction, on real-world attributed networks. The qualitative analysis from visualization and query task also validates our approach.

The rest of the paper is organized as as follows. Section 2 discusses the related work. In Sect. 3 we introduce some definitions and formally describe the problem. In Sect. 4 we present our proposed framework GAT2VEC. We describe the experimental setup and experimental results in Sect. 5, before concluding in Sect. 6.

2 Related work

The embedding of a graph can be learned using the structure, the attributes of nodes, or a combination of them. Based on the network structure, various approaches [19,21,25,26] have been proposed to learn network representations that preserve the structural proximities of all nodes. Tang et al. [25] generates the representation from the top-d eigenvectors of the modularity matrix of the graph; later they proposed SOCIODIM [26], which generates an embedding from the d-smallest eigenvectors of the normalized Laplacian matrix. Most recently, deep learning methods have been used

to learn embeddings. In their seminal paper, Perozzi et al. [19] proposed DEEPWALK, a generalized SKIPGRAM model [16] to learn a representation of a graph. SKIPGRAM is a neural network model for learning representation of words in text. DEEPWALK generates a set of random walks for each vertex, that are provided as input to SKIPGRAM as if they were sentences. Tiag et al. [24] proposed the LINE, which preserves both the local and global network structure by capturing first-order proximity as well as second-order proximity. The second-order proximity is defined in terms of the number of shared neighbors between two vertices. Grover et al. [9] pointed out that a vertex has two relations with other nodes in a network, namely homophily and structural equivalence. Vertices that have similar roles in a network (e.g., hubs) have structural equivalence. Based on this observation, NODE2VEC has been proposed [9]. In NODE2VEC, depth first search (DFS)—or breadth first search (BFS)-biased random walks are performed to capture either the homophily or the structural equivalence, respectively.

Another possible approach to learn an embedding of a graph is to use the content/attributes of nodes and omit the structure of graph. On the content information, various state-of-art text-based approached can be used, such as topic models like LDA [1], deep neural networks like SKIPGRAM [16] or the paragraph vector model introduced by Le and Mikolov [14].

When leveraging the heterogeneous information from both the structure and the attributes, the learned representation tend to be more accurate. The attributes enable to exploit the contextual information in order to learn embeddings for sparsely connected or even disconnected nodes. While the structure enables to maintain the structural proximity of nodes in learned embeddings, the two sources of information complement each other in learning the precise low-dimensional embedding of nodes. Yang et al. [29] showed that DEEPWALK is equivalent to factorizing an adjacency matrix M and proposed a model called TADW which incorporates text features of nodes by factorizing a text-associated matrix. Despite its promising results, TADW has several limitations: (1) it factorizes an approximate matrix, hence, the representation is poor; (2) it highly depends on computationally expensive singular value decomposition (SVD); (3) it does not scale-up to large graphs. Kefato et al. [12] proposed to use diffusion event information of a network besides structure and content information to learn a representation which preserves the diffusion context besides structural and content contexts of vertices.

Various semi-supervised approaches [11,18,23,30] have been proposed, using label information to learn an embedding. The labels are the classes associated with the vertices and are used in training the classifier on learned embeddings for tagging the non-labeled nodes. The results show that by incorporating the labels in the learning process, the embeddings tend to be precise. The intuition behind using label information is that nodes with similar labels tend to have strong interconnections and high similarity of attributes, thus, should be embedded closely. Pan et al. [18] proposed tri-party deep network representation (TRIDNR) to use three sources of information: structure, text, and partial labels for embeddings. TRIDNR uses two layers of neural network: one based on DEEPWALK to learn a representation based on structure, while the second layer employs DOC2VEC [14] to learn a representation on content and partial labels. The final representation is a linear combination of the two. Other approaches, such as PLANETOID [30] use only labels along with structure in learning

an embedding. PLANETOID proposed the use of both transductive and inductive models for jointly predicting the class label and neighborhood contexts.

The above mentioned approaches work with homogeneous networks. Recent works [6,23] learn the representation of nodes in heterogeneous networks. Dong et al. [6] proposed METAPATH2VEC, which uses the metapath-based random walks to generate the semantic relationship between different kind of nodes in the network. However, the work ignores the relationships between similar nodes, such as citations between papers. Predictive text embedding (PTE) [23], a semi-supervised approach embeds a heterogeneous text network which include words, documents and labels. Recent advances in NRL methods include convolutional networks [13] and inductive learning-based approaches [10].

3 Problem definition

We consider an *attributed graph* $G = (V, E, A)$, composed of a set of vertices V, a set of edges E, and an *attribute function* $A : V \rightarrow 2^{\mathbb{A}}$, where $\mathbb{A}$ is the set of all possible *attributes*.

Consider for example a citation network: nodes could be papers, directed edges could be citations among them, while attributes could be the title, the keywords, topics, etc.

The objective of network representation learning is to embed vertices in a low-dimensional space where the graph properties such as pairwise relationships between vertices, the structure of vertex local neighborhood, and the attribute similarities of vertices in the input space are preserved. The properties between vertices can be captured through local information (e.g., followers, citations, friendship relations) or global information (e.g., h-hop neighborhood, community affiliation). The similarity of a vertex with respect to other vertices represents its contextual information. The contexts obtained using structural information of the network are called *structural contexts*.

Likewise, we can obtain *attribute contexts* based on the attributes of the nodes. The contextual information based on attributes defines the semantic relationship between vertices. For example, in a citation network, two papers having similar keywords share contextual information irrespective of their distance in structure.
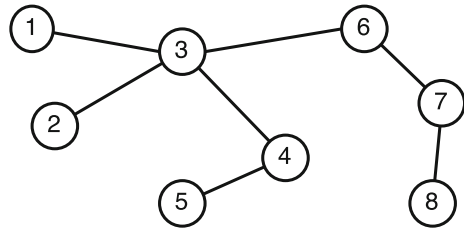
We investigate the problem of integrating structural and attribute contextual information obtained from a partially attributed graph and employ a neural network model to jointly learn a representation of the vertices in a low-dimensional space. Informally, the problem can be described as follows:

3.1 Problem

Given an attributed graph G as shown in Fig. 1, we aim at learning a low-dimensional network representation $\Phi \colon V \rightarrow \mathbb{R}^d$, where $d \ll |V|$ is the dimension of the learned representation, such that the structural and attribute contextual informations are preserved.

The learned representation Φ is generic and can provide feature inputs for various machine learning tasks, such as classification and link prediction. Its validity is eval-

uated through such machine learning tasks; the precise figure of merits to be used are described in the respective Sects. 5.4 and 5.5. For example, if the learned representation is able to classify vertices with high precision and recall (combined as F1 score), that means that the contextual information of vertices is well-preserved.

In this paper, we evaluate our approach on classification of vertices (multi-class and multi-label) and link prediction. We also perform a qualitative analysis (nearest-neighbor search and visualization).

In this paper, we consider G as a homogeneous, un-weighted and partially attributed graph.

4 GAT2VEC framework

In this section, we present a detailed description of our proposed framework GAT2VEC[1] (Fig. 4) which learns a network embedding from structural and attribute information. For each vertex, we obtain its structural and attribute contexts with respect to other vertices through random walks. Then, we integrate these two contexts to learn an embedding which preserves both structural and attribute proximities. The GAT2VEC method is outlined in Algorithm 1. Specifically, our framework consists of three stages:

– Network generation.
– Random walks.
– Representation learning.

4.1 Network generation

From the attributed graph, G, we obtain two graphs:

1. A connected structural graph $G_s = (V_s, E)$, consisting of a subset $V_s \subseteq V$ of vertices that have connections included in the set E of edges. We refer to vertices in V_s as *structural vertices*. An edge $(p_s, q_s) \in E$ encodes a structural relationship between nodes as shown in Fig. 2.
2. A bipartite graph $G_a = (V_a, \mathbb{A}, E_a)$, consisting of (i) the subset of *content vertices* $V_a \subseteq V$ that are associated with attributes, (ii) the set of possible *attribute vertices* $\mathbb{A}$ as given in the definition of *attributed graph* in Sect. 3, and (iii) the set of

---

Fig. 3 A bipartite graph between content nodes and attributes
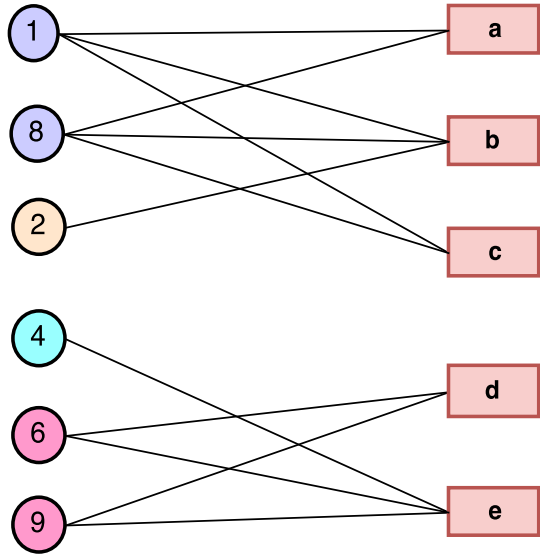
edges E_a connecting content vertices to the attribute vertices that are associated by function A:

$$V_a = \{v : A(v) \neq \emptyset\}$$
$$E_a = \{(v, a) : a \in A(v)\}$$

Proposition 1 *Two content vertices $u, v \in G_a$ are reachable only via attribute vertices.*

Following Proposition 1, the path between content vertices contains both content and attribute vertices. Therefore, the content vertices contained in the path have a contextual relationship because they are reachable via some attribute vertices. These content vertices form the attribute contexts. The intuition behind our approach is "two entities are similar, if they are connected with similar objects". This phenomenon can be observed in many applications, e.g., in the bipartite "user-product" graphs, where "users" buy "products" and two users are similar if they buy similar products.

Such bipartite network structure has also been used in [23] which models a network between documents and the words present in them.

4.2 Random walks

Informally speaking, similarity between entities could be measured in several ways. For example, it could be measured based on the distance of two nodes in the graphs. In the attributed graph of Fig. 3, e.g., nodes 2 and 8 are both one attribute away from node 1, and thus we could say that they have equal similarity to 1; but there are three
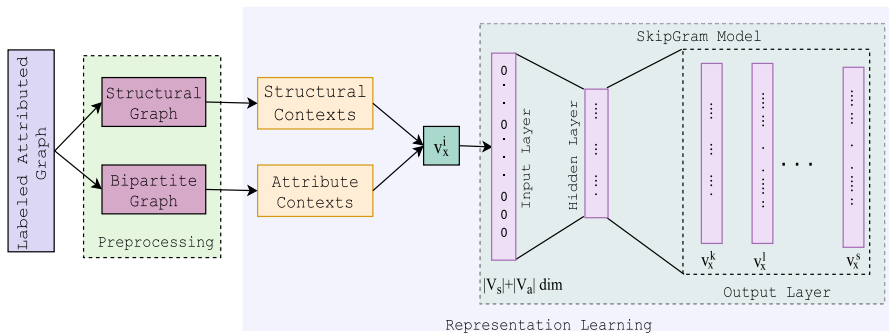
Fig. 4 The architecture of GAT2VEC

paths connecting 1 and 8, while there are only two paths connecting 1 and 2; thus, 1 and 8 are more similar than 1 and 2.

While several approaches have been used [9,18,19,24,27], we adopt short random walks to obtain both the structural and the attribute contexts of vertices at the same time. The short random walks enable to effectively capture the contexts in which nodes have high similarity [19].

Random walks are performed on both G_s and G_a. The random walks over G_s capture the structural context. For each vertex, γ_s random walks of length λ_s are conducted to build a corpus R. This contextual information is used in the embedding, with the aim of maintaining the local and global structure information. We denote r_i as the i-th vertex in the random walk sequence $r \in R$.

For example, a random walk in the graph of Fig. 2 could be the following: $r = (2, 3, 4, 3, 1)$, with length length 5, starting at vertex 2 and ending at vertex 1.

In the bipartite graph G_a, a random walk starts with a content vertex and jumps to other content vertices via attribute nodes. The attribute vertices act as bridges between content vertices and help in determining the contextual relationships among them, i.e., which content vertices are closely related. As we are interested in how often such vertices co-occur in random walks and not in which attributes have been traversed to connect them, we have omitted the attributes in our random walks. Thus, the walks contain only vertices from V_a. Group of vertices that have high similarity in attributes are likely to appear frequently together in the random walks. Similar to G_s, we perform γ_a random walks of length λ_a and build a corpus W, and w_j specifies the j-th vertex in random walk sequence $w \in W$.

For example, a random walk with attributes in the graph of Fig. 3 could be the following: $[2, b, 1, c, 8, b, 2, b, 8]$. Since we are skipping attribute nodes in walks, therefore the corresponding walk is $w = [2, 1, 8, 2, 8]$, with length 5, starting from vertex 2 and terminating in vertex 8.

4.3 Representation learning

The architecture of GAT2VEC is shown in Fig. 4. The input is given by structural and attribute contexts obtained from the respective graphs (line 11, 12 of Algorithm 1). We

---

Algorithm 1: The GAT2VEC Algorithm

    Input   : An Attributed Graph $G = (V, E, A)$
    Output   : $\Phi: V \to \mathbb{R}^d, d \ll |V|$
    Parameters : walks per node γ_s, γ_a;
               length of walks λ_s, λ_a;
               context window size c;
               embedding size d
1 Obtain G_s and G_a from G
2 **for** $v \in V_s \cup V_a$ **do**
3 | initialize $\Phi(v)$ //initializing the vectors
4 **end**
5 **for** $u \in V_s$ **do**
6 | $R_s(u) = \text{RandomWalks}(G_s, u, \gamma_s, \lambda_s)$
7 **end**
8 **for** $v \in V_a$ **do**
9 | $W_a(v) = \text{RandomWalks}(G_a, v, \gamma_a, \lambda_a)$
10 **end**
11 $R = \oplus_{u \in V_s} R_s(u)$
12 $W = \oplus_{v \in V_a} W_a(v)$
13 $\Phi = SkipGram(R, W, c)$ //as per Eq. 1
14 return Φ

---

use the SKIPGRAM model to jointly learn an embedding based on these two contexts. From each context, structural or attribute, a vertex $v_x \in V_s|V_a$ is selected and is input to SKIPGRAM. The input vertex is one-hot encoded vector $\{0, 1\}^{|V_s \cup V_a|}$. The input vertex v_x is the target vertex and the output layer produces the $2c$ multinomial distributions of associated context vertices to the given input vertex. c is the context size i.e., the number of predicted vertices before or after the target vertex. Likewise, the output vertices can belong either to structural vertices or to attribute vertices, or to both, depending on their co-occurrence in random walks.

Given a target vertex, the objective of GAT2VEC model is to maximize the probability of its structural and attribute contexts. Similar to previous studies [9,19], we follow the assumption that given a target vertex, the probability of a context vertices are independent of each other. Therefore, learning is described by the following objective function:

$$L = \sum_{r \in R} \sum_{i=1}^{|r|} \log p(r_{-c} : r_c | r_i) + \sum_{w \in W} \sum_{i=1}^{|w|} \log p(w_{-c} : w_c | w_i) \tag{1}$$

The Eq. 1 can be written as:

$$L = \sum_{r \in R} \sum_{i=1}^{|r|} \sum_{\substack{-c \le j \le c \\ j \neq i}} \log p(r_j | r_i) + \sum_{w \in W} \sum_{i=1}^{|w|} \sum_{\substack{-c \le t \le c \\ t \neq i}} \log p(w_t | w_i) \tag{2}$$

where $r_{-c} : r_c$ and $w_{-c} : w_c$ correspond to a sequence of vertices inside a contextual window of length $2c$ in random walks contained in corpus R and W, respectively.

The first term uses the structural contexts, while the second is for learning from attribute contexts. If $|V_a| = 0$, then the model will become DEEPWALK, i.e., learns only from structure. The term $p(r_j|r_i)$ is the probability of the j-th vertex when r_i is the central vertex in the structural context r, while the term $p(w_t|w_i)$ is the probability of the tth vertex when w_i is the central vertex in the attribute context w. These probabilities can be computed using the softmax function. The probability $p(r_j|r_i)$ can be computed as:

$$p(r_j|r_i) = \frac{\exp\left(\varphi(r_j)^T \Phi(r_i)\right)}{\sum_{v_s \in V_s} \exp\left(\varphi(v_s)^T \Phi(r_i)\right)} \tag{3}$$

where $\varphi(\cdot)$, $\Phi(\cdot)$ are representations of a vertex when it is considered as a context vertex or a target vertex respectively. Similarly, we can compute $p(w_t|w_i)$ following Eq. 3.

The softmax calculation is computationally expensive due to normalization over all vertices of a graph. Thus, we approximate it by using the hierarchical softmax function [17]. Following [14], we used Huffman coding to build binary trees for hierarchical softmax which has vertices as leaves. Therefore, in order to compute the probability, we just need to follow the path from the root to the leaf node of the tree. Thus, the probability of a leaf node r_j to appear in the structural context is:

$$p(r_j|r_i) = \prod_{h=1}^{d} p(s_h|\Phi(r_i)), \tag{4}$$

where $d = \log|V_s|$ is the depth of the tree and s_h are the nodes in the path with $s_o =$ being the root and $s_d = r_j$. Furthermore, modeling $p(r_j|r_i)$ as a binary classifier reduces the computational complexity to the order of $O(\log(|V_s|))$. The same can be applied to compute the probability for vertices in attribute contexts. Given that we are computing the probabilities from two contexts, this leads to the overall computational complexity of $O\left(\log|V_s| + \log|V_a|\right)$.

4.4 GAT2VEC-WL

In our work, we can also exploit the labels for precise learning of an embedding in a semi-supervised manner. The idea behind this approach is that the vertices sharing labels are similar and thus should be encoded close to each other in the embedding. We propose GAT2VEC- WL to incorporate labels associated with content vertices as attributes of vertices. The labels will be defined as an attribute vertex in the bipartite graph defined in Sect. 4.1. The labels will be helpful in generating the contexts in which the vertices sharing labels will appear together.

Since real-world networks are partially labeled, therefore, we randomly pick a percentage L_P of labeled content vertices and incorporate their labels as attributes. After learning an embedding, these selected vertices are used in subsequent machine

Table 1 Dataset statistics

Dataset	$|V|$	$|E|$	#Lbls	#Lbl vertices	V_s	V_a	$|\mathbb{A}|$	$|E_a|$
DBLP	60,744	52,890	4	60,744	17,725	60,720	8618	356,230
CITESEER	38,996	77,218	10	10,310	36,227	10,107	3986	59,477
BLOGCATALOG	70,004	1,409,112	60	70,004	55,771	57,709	5413	269,363

learning tasks. For example, for classification, we train our classifier on these selected vertices and predict labels for the rest.

5 Experiments

In this section, we provide an overview of widely used datasets and discuss the details of the experimental setup to compare the performance of our proposed approach, GAT2VEC, against a collection of state-of-the-art approaches. We will validate the representation learned so far to perform two important tasks, namely vertex classification and link prediction.

5.1 Datasets

We use three real-world datasets: the social network BLOGCATALOG [2] and the citation networks DBLP [5] and CITESEER [4]. Table 1 summarizes their statistics, including information about the associated structural and attribute graphs.

DBLP and CITESEER are widely used for experimentation [18]. DBLP contains bibliographic data in computer science, extracted from a selected list of 34 conferences from 4 research areas. CITESEER is a multi-disciplinary dataset consisting of papers from 10 research fields.

In the DBLP and CITESEER datasets, the title of the paper constitutes the vertex content. We pre-process these titles to remove stop words. From each dataset, we selected the words occurring at least *th* times as vertex attributes. At *th* = 3, DBLP, CITESEER and BLOGCATALOG have 8618, 3986, and 5413 attributes, respectively.

BLOGCATALOG [28] is a social network of bloggers. The labels represent the interests of bloggers and each blogger may be associated to multiple labels. The attributes are keywords generated from users blog.

5.2 Baseline methods

We compare GAT2VEC against state-of-the-art NRL algorithms for learning graph embeddings: two structure-based methods (NODE2VEC and DEEPWALK), one content-based (DOC2VEC), and two methods using both structure and content (TRIDNR and TADW). We also consider variants of TRIDNR and TADW. All the results in this paper are obtained using the code released by the authors.

- NODE2VEC [9] uses biased random walks to generate the vertex sequences based on in-out parameters p, q. For each of our datasets, we learned these hyper-parameters as described in the original paper.
- DEEPWALK [19] is an approach based on random uniform walks to learn a d-dimensional feature representations of vertices in a network using only structure information.
- DOC2VEC [14] is an unsupervised neural network model to learn a representation for variable length texts such as sentences, paragraphs or documents. The model learns both word vectors and document vectors. We fed the text associated with each vertex to the model and obtained a representation for each vertex.
- TRIDNR [18] is a learning model which uses three sources of information: network structure, vertex content, and label information to learn the representation of vertices. TRIDNR uses two models: DEEPWALK to learn the representation from structure, and DOC2VEC to capture the context related to node content and label information. The final representation is a linear combination of outputs of these two models.
- TRIDNR NOLBL is an unsupervised variant of TRIDNR which learns a representation without using vertex labels.
- TADW [29] learns a d-dimensional representation of vertices by factorizing a text-associated matrix. Before factorizing the matrix, the features are reduced using SVD.
- TADW NOSVD is a modified version of TADW in which we fetched raw feature vectors of vertices without performing SVD on them to learn a representation of a network.
- GAT2VEC- WL is a semi-supervised variant of GAT2VEC to learn a representation using labels as attributes.
- GAT2VEC- BIP is another version of GAT2VEC which learns a representation on the bipartite graph only. Thus, no structural information is used for learning.

5.3 Experimental setup and parameter settings

We perform multi-class classification on DBLP and CITESEER, and multi-label classification on BLOGCATALOG using the respective learned representation.

We set the parameters of GAT2VEC as follow: number of walks, $\gamma_s = 10$ and $\gamma_a = 10$; walk length, $\lambda_s = 80$ and $\lambda_a = 80$. The representation size is $d = 128$; window size is $c = 5$, the threshold th is equal to 3. For fairness of comparison, the parameters that are in common between GAT2VEC and the other methods are set with the same value, while the rest of the parameters are set to their optimal default values as reported in the respective papers.

5.4 Vertex classification

For classification, we used one-vs-rest logistic regression classifier LIBLINEAR [7] with default parameters for training the data, and then predict the unlabeled vertices. We

Table 2 Multi-class classification on DBLP

Metric	T_R (%)	DEEPWALK	NODE2VEC	DOC2VEC	GAT2VEC- BIP	TADW	TADW NOSVD	TRIDNR NOLBL	GAT2VEC
MICROF1	10	51.8	48.9	74.2	76.5	68.4	60.8	74.3	*79.0*
	30	52.5	49.3	76.1	77.4	68.8	61.0	74.8	*79.4*
	50	52.7	50.0	78.0	77.6	69.0	61.1	75.1	*79.5*
MACROF1	10	41.0	44.7	67.2	70.1	62.9	56.1	67.0	*72.6*
	30	42.1	45.4	70.0	70.8	63.1	56.5	67.7	*73.1*
	50	42.3	45.7	71.2	71.1	63.5	56.5	68.2	*73.4*

Italic values indicate the best results obtained from different algorithms

Table 3 Multi-class classification on CITESEER

Metric	T_R (%)	DEEPWALK	NODE2VEC	DOC2VEC	GAT2VEC- BIP	TADW	TADW NOSVD	TRIDNR NOLBL	GAT2VEC
MICRO- F1	10	44.0	63.7	60.9	65.5	49.0	32.5	63.3	*66.3*
	30	47.8	66.5	70.0	70.2	51.2	35.2	67.6	*70.7*
	50	49.0	67.3	*72.8*	71.4	51.8	36.0	69.8	72.0
MACRO- F1	10	35.1	54.6	55.5	61.4	44.0	28.4	57.8	*62.2*
	30	37.9	56.3	66.4	66.5	45.8	30.5	62.0	*67.1*
	50	39.0	57.0	*68.8*	67.8	46.4	31.4	64.5	68.5

Italic values indicate the best results obtained from different algorithms

randomly selected a sample of size $T_R \in \{10, 30, 50\%\}$ of vertices as training set to train the classifier and used the rest as a test set.

In case of GAT2VEC- WL, we randomly selected a percentage of labeled vertices, $L_P \in \{10, 30, 50\%\}$ and incorporated their values as attributes for learning a representation. Then these labeled vertices are used in training the classifier and predicting the labels of the remaining vertices.

The metrics used to compare our approach against the baselines are the widely used MICRO- F1 and MACRO- F1 scores. For fairness in comparison, we used the same training set of GAT2VEC across all baselines for training the classifier. For each training ratio, we repeat the process 10 times and report the average scores.

5.4.1 Results and discussion

Tables 2, 3 and 4 show the MICRO- F1 and MACRO- F1 results on the DBLP, CITESEER and BLOGCATALOG datasets. The results are consistent with the results reported in the baselines. GAT2VEC outperforms all baseline methods in all three datasets, even when a small number ($T_R = 10\%$) of vertices are used for training. This makes GAT2VEC suitable in real-world, sparse-labeled datasets.

The results validate our approach of generating structural and attribute contexts and then subsequently learn the representation from both of them. In fact, GAT2VEC beats the state-of-the-art, attribute-based method TRIDNR in all three datasets. This implies that properly modeling attribute information increases the preciseness of embeddings.

Table 4 Multi-label classification on BlogCatalog

Metric	T_R (%)	DeepWalk	Node2Vec	Doc2Vec	GAT2Vec- bip	TADW	TADW noSVD	TriDNR noLbl	GAT2Vec
Micro- F1	10	29.8	31.0	40.3	47.3	38.0	24.8	34.2	*49.5*
	30	31.8	32.5	41.4	49.2	38.8	25.4	36.3	*51.2*
	50	32.2	32.8	41.9	49.6	39.0	25.9	36.8	*51.6*
Macro- F1	10	16.2	17.0	23.4	34.0	23.4	8.1	21.0	*36.3*
	30	17.8	18.6	25.8	36.6	24.3	8.2	21.5	*38.8*
	50	18.6	18.8	26.6	37.0	24.6	8.5	22.2	*39.3*

Italic values indicate the best results obtained from different algorithms

Furthermore, GAT2Vec outperforms TADW in all datasets. This is due to the structural and attribute sparsity. The impact of attribute sparsity can be ascertained from results of CiteSeer dataset which is much more sparse than dblp, as only 10,310 out of 38,996 vertices have titles associated with them. This is also the reason of deviation from reported results in TriDNR which learns a representation using only 10,310 vertices. The other reason for this poor performance is due to using an approximation of DeepWalk. The dismal performance of TADW noSVD shows high dependence of the TADW approach on the costly SVD method; without SVD, the sparsity issue is aggravated as shown in Table 3 for the CiteSeer dataset.

GAT2Vec- bip outperforms the content-based method Doc2Vec in BlogCatalog dataset, but has almost similar performance in case of CiteSeer and dblp datasets. This *highlights* the appropriate modeling of attribute information even in cases when attribute information is non-cohesive and semantically unrelated such as in BlogCatalog dataset.

In the case of CiteSeer and dblp datasets, Doc2Vec uses only titles of papers which has rich information as compared to the structure. Thus, it performs better than DeepWalk but has comparable performance with Node2Vec as it exploits the structural information much efficiently than DeepWalk. Furthermore, in BlogCatalog dataset, Doc2Vec under-performs as the contents do not contain rich information. In this case, our modeling proves to be better as shown by results of GAT2Vec- bip.

Structure-based methods such as DeepWalk and Node2Vec perform poorly, justifying the use of information from multiple sources to learn embeddings.

Table 5 shows the Macro- F1 of the three datasets when the labels are incorporated in learning the embedding. We have included just TriDNR as it is the only baseline which uses labels and attributes in learning. The results of GAT2Vec prove our claim that including the labels increases the quality of embedding in classification task. Furthermore, it also implies that our proposed approach of incorporating attributes is sufficient to generate the proper contexts. GAT2Vec has superior performances with respect to TriDNR, which in turn implies that performs better than the rest of the baselines.

5.5 Link prediction

One important network analysis task is link prediction, and in the following we present the setting and comparison of GAT2Vec and the baselines for this task. We follow a

Table 5 MACRO- F1 score of classification (using labels)

Dataset	Tr (%)	TRIDNR	GAT2VEC- WL
DBLP	10	69.5	*75.2*
	20	72.0	*81.4*
	30	72.2	*86.3*
CITESEER	10	64.0	*79.2*
	20	71.1	*77.0*
	30	73.8	*83.0*
BLOGCATALOG	10	21.0	*39.7*
	20	23.8	*53.0*
	30	24.9	*63.8*

Italic values indicate the best results obtained from different algorithms

similar strategy as [9,27] and sample 15% of the observed or true edges ($T_E \subset E$) and remove them from the graph while ensuring the residual network is connected. We also sample 15% false edges ($F_E \cap E = \emptyset$). Next, each network representation algorithm is trained using the residual network. Once the training is complete, we get an embedding for each node. Then we predict the probability of an edge (u, v) being a true edge, for all edges $(u, v) \in T_E \cup F_E$ that we have just sampled, according to the following equation.

$$p(u, v) = \frac{1}{1 + \exp^{-\Phi(u) \cdot \Phi(v)}} \tag{5}$$

We finally rank edges according to the probability p and consider the top-k ones in the ranking at different values of k and measure the prediction performance using precision at k, $P(k)$. Let R be set of edges ranked according to p and R_k be the top-k elements of R. Then, $P(k)$ computes the fraction of correctly predicted edges from the top-k predictions, and it is specified by

$$P(k) = \frac{|T_E \bigcap R_k|}{|R_k|} \tag{6}$$

5.5.1 Results and discussion

The results for link prediction is given in Tables 6 and 7. One can notice that for the DBLP dataset (Table 6), where the structural network is very sparse, algorithms which are based only on the structural knowledge produce poor result. Our algorithm, which alleviate the structural sparsity by considering connections with attributes, significantly outperforms the baselines including those that integrate attribute and label information. Our finding also shows that when the structural network is sufficiently dense (BLOG-CATALOG), structure based algorithms could be sufficient in link prediction as shown in Table 7.

Table 6 $P(k)$ for link prediction on DBLP

	DEEPWALK	NODE2VEC	TADW	TRIDNR	GAT2VEC
P(1000)	0.93	0.92	*1.0*	0.95	0.99
P(5000)	0.67	0.73	0.80	0.70	*0.93*
P(10000)	0.54	0.59	0.54	0.52	*0.68*
P(15000)	0.43	0.45	0.41	0.42	*0.48*
P(20000)	0.34	0.34	0.33	0.33	*0.37*
P(25000)	0.27	0.27	0.37	0.38	*0.30*

Italic values indicate the best results obtained from different algorithms

Table 7 $P(k)$ for link prediction on BLOGCATALOG

	DEEPWALK	NODE2VEC	TADW	TRIDNR	GAT2VEC
P(1000)	0.85	*0.92*	0.20	0.81	0.74
P(5000)	0.70	*0.71*	0.17	0.89	0.64
P(10000)	*0.66*	0.65	0.18	0.51	0.63
P(15000)	0.63	0.63	0.19	0.49	0.63
P(20000)	0.62	0.61	0.20	0.48	*0.63*
P(25000)	0.61	0.60	0.21	0.47	*0.62*

Italic values indicate the best results obtained from different algorithms

5.6 Parameter sensitivity

GAT2VEC requires various parameters to create G_s and G_a, and learn an embedding. The sensitivity of the choice of parameters on structural networks have been investigated in previous works, such as [9,19]. We investigate the parameter sensitivity of γ_a, λ_a, and *th* on attributed graph. Furthermore, we analyze the joint parameter sensitivity of number of walks (γ_s, γ_a), and walk length (λ_s, λ_a) on G_s and G_a. In these experiments, we evaluate on MICRO- F1 under the same evaluation process given in Sect. 5.4. All other parameters are set to default values unless mentioned.

5.6.1 Impact of γ_a and λ_a

Figure 5a, b shows the effects of varying the number of walks per vertex and the walk length on bipartite attribute graphs. We empirically observe that these parameters have direct proportionality relationship with the sparsity. In a very dense graph, a small number of short walks are adequate to explore the neighborhood to capture the contexts. DBLP is highly dense, while CITESEER is very sparse, as shown in Fig. 6. Therefore, for DBLP, our framework is able to generate precise representations at lower values of γ_a and λ_a. Increasing the number of walks and the walk length has detrimental effect on the learning representation as it explores the large parts of the graph, which in turn generates contexts that are noisy i.e., include less significant vertices. In case of CITESEER, we see an opposite trend: precise representations are learned for higher values of γ_a and λ_a. This is motivated by the sparsity of the dataset, which is composed
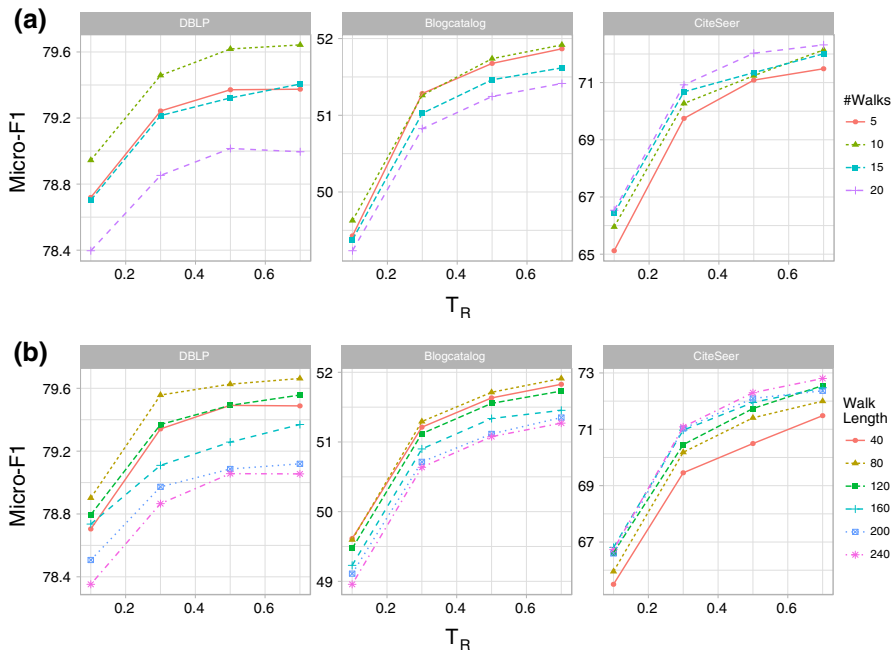
Fig. 5 G_a Parameter sensitivity on: **a** number of walks (γ_a), **b** walk length (λ_a)

of 10 different disciplines, thus requiring a larger number of longer walks to explore the graph adequately and generate the appropriate contexts.

5.6.2 Impact of γ_s, γ_a, and λ_s, λ_a

Here we analyze the joint parameter sensitivity on the DBLP dataset. In Fig. 7(a) we increase the number of walks (γ_s, γ_a) jointly on both G_s and G_a, while keeping walk length to 10. In second case 7(b), walk lengths are jointly increased, while the number of walks is kept constant to 10. As discussed in previous case, increasing the number of walks or walk length gathers more contextual information, thus, learning more precise representations. But, an excessive number of walks and large walk lengths are detrimental as it generates noisy contexts which lead to poor representation, as shown in case $\gamma_s = \gamma_a = 25$ (Fig. 7a), and $\lambda_s = \lambda_a = 240$ (Fig. 7b).

5.6.3 Impact of th

We have shown the usefulness of attributes (Tables 2, 3, 4) for learning high quality embeddings. We further stress upon the importance of attributes by varying the parameter *th*. It is pertinent to mention that increasing the value of *th* implies a decrease in the number of attributes. From Fig. 8, it is evident that as we decrease the number of attributes, the accuracy of embeddings decreases. At *th* = 1, the performance is lower due to the noise introduced by including all possible words as attributes.
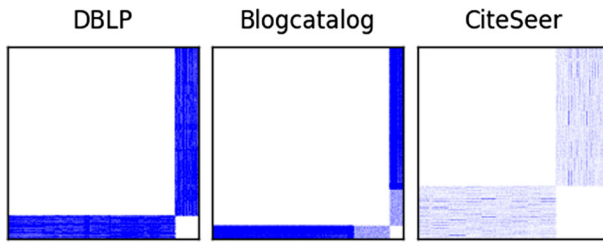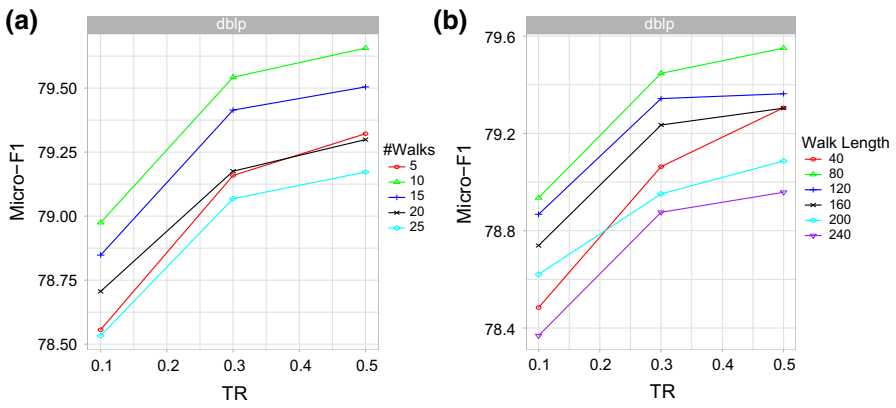
Fig. 6 Sparsity of attributed graph G_a

Fig. 7 G_a Joint parameter sensitivity on: **a** number of walks (γ_a), **b** walk length (λ_a)
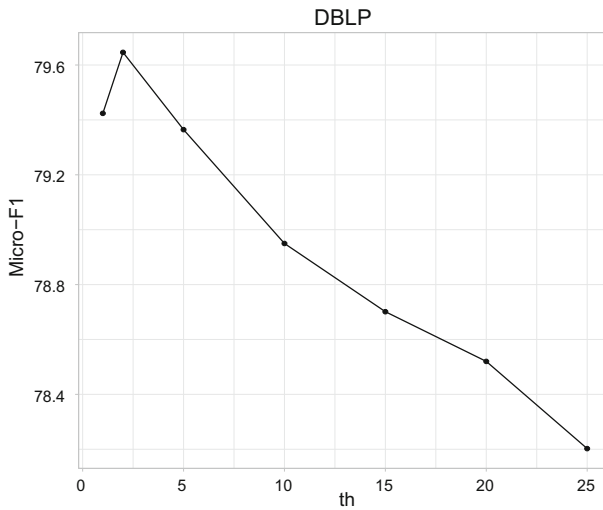

Fig. 8 Effect of parameter-*th*

Table 8 Nearest neighbor top 3 results

Algorithm	Query: group formation in large social networks: membership, growth, and evolution	Cites	#Cit.
GAT2VEC	1. Microscopic evolution of social networks	Yes	4
	2. Structure and evolution of on-line social networks	No	7
	3. Maximizing the spread of influence through a social network	Yes	14
TRIDNR	1. Searching for rising stars in bibliography networks	Yes	0
	2. A framework for community identification in dynamic social networks	Yes	9
	3. A framework for analysis of dynamic social networks	Yes	2
TADW	1. Tutorial summary: large social and information networks: opportunities for ML	No	0
	2. Exploring and visualizing academic social networks	No	0
	3. Seeing sounds: exploring musical social networks	No	0
NODE2VEC	1. Characterizing and predicting community members from evolutionary and heterogeneous networks	Yes	0
	2. Searching for rising stars in bibliography networks	Yes	0
	3. Constant-factor approximation algorithms for identifying dynamic communities	Yes	0

5.7 Qualitative analysis

To evaluate qualitatively the learned representations from our proposed approach, we performed nearest neighbor search and visualization of the learned embeddings.

5.7.1 Case study

We carried out a case study on the DBLP dataset, by selecting a query paper and then obtaining the three nearest neighbors with respect to its representation, using the cosine distance metric. We generated an embedding through GAT2VEC- WL, with 10% vertices labeled. The query paper is "Group formation in large social networks: membership, growth, and evolution", which has the highest number of citations in DBLP (1666 on July 2017). The paper studies the effect of network structure on the evolution of communities in social networks. Therefore, the results of the nearest neighbors should have attribute contexts related with social networks, communities, and evolution. In addition, the resulting papers should also preserve its structural contexts. Table 8 lists the query results on different embeddings, along with citation relationship, and common citations between the query and the result.

The results returned by GAT2VEC are relevant in attribute contexts to the query. The first and third results are cited by the query, which shows that GAT2VEC preserves the direct proximity. Indirect proximities are preserved as well, as it evidenced by the second query result which is not cited by the query but have 7 common cited papers.
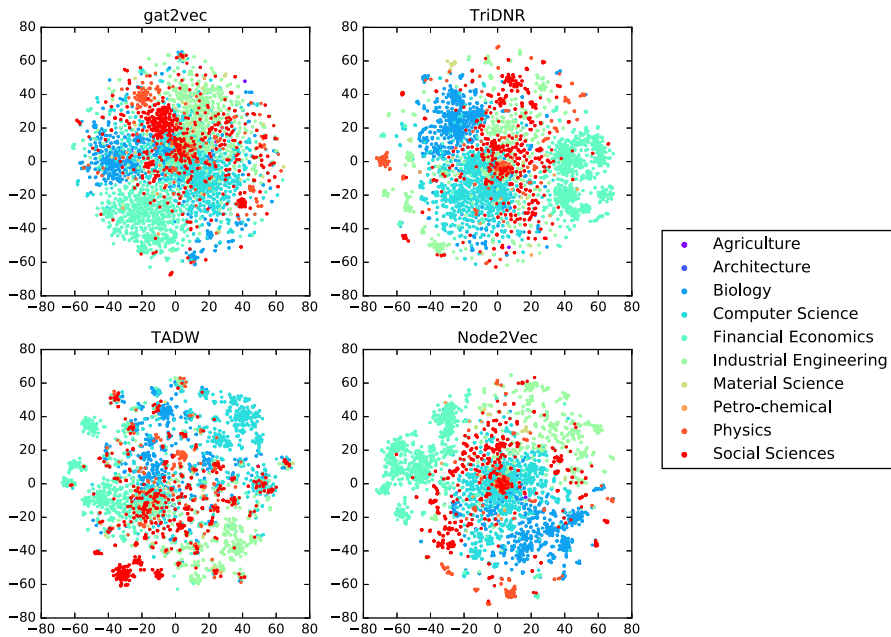
Fig. 9 2-D t-SNE projection of CITESEER dataset

Therefore, our proposed method GAT2VEC generates accurate structural and attribute contexts which eventually help to learn precise embeddings.

Apart from the first one, the results from TRIDNR are quite related with the query as they are either directly cited or have common citations with query paper. The results from TADW have some relevance with the query. But structurally, there is neither citation relationship nor any common cited papers with the query paper. The possible reason for such poor results could be due to matrix approximation for obtaining structural contexts. NODE2VEC results show what happens when ignoring the attribute information, nonetheless NODE2VEC exploits the structural information as the results and query have a direct relationship.

The above qualitative analysis support the claim that using multiple sources of information aids in learning the precise embeddings.

5.7.2 Network visualization

The learned embedding can be projected in a two dimensional plane to visualize the co-relationship between the nodes. We use the learned embedding from TRIDNR, TADW, and NODE2VEC on the CITESEER dataset for comparison. Papers which cite less than five papers are filtered out. Since these embeddings are in a higher dimensional space, we use t-SNE [22] to reduce these to 2-D space. The visualizations of different approaches are given in Fig. 9.

The visualization using TADW is not very meaningful; in fact, the papers belonging to *Social Sciences, Biology* are not clustered. A small dispersion of *Social Sciences*

papers is acceptable as they are cross-cited across different fields such as *Computer Science*. Unfortunately, no proper cohesive group is formed here. The problem is related to the use of the approximation approach for random walks. The results from NODE2VEC are much better, as it depicts some communities properly (*Financial Economics, Computer Science*). This is because NODE2VEC performs biased walks which are effective in capturing the structural equivalences.

The TRIDNR performs quite well as some clearly defined clusters can be seen compared to other two approaches. It still does not learn meaningful embeddings as papers from *Social Sciences* and *Industrial Engineering* are not clustered but are dispersed. Our proposed approach GAT2VEC performs better than all given methods. The clusters are well formed and depict the overlapping properly. The papers from *Social Sciences* can be seen forming a well-defined cluster in contrast to other approaches. In addition to it, this cluster is close to *Computer Science* and *Financial Economics* clusters, which is plausible as there are cross citations between these fields. Thus, our proposed approach learns an embedding which preserves the structural and attribute equivalence in the underlying graph.

6 Conclusion

In this paper, we propose GAT2VEC, which uses both network structure and vertex attributes to learn a representation of an attributed graph. GAT2VEC uses a novel method to acquire attribute contexts. We extract the structural contexts from the network, while the attribute contexts are obtained from a bipartite graph of vertices and attributes. We employ a shallow neural network model to jointly learn a representation from the two contexts. The paper shows that by properly modeling multiple sources of information associated with the network and by employing an appropriate learning methodology, the learned representation can preserve as much information as the original input graph. The extensive experiments on real-world datasets show that our approach can learn precise representation of structural and attributes contexts of vertices in the graph. This is corroborated by the significant gains over the state-of-art baselines in vertex classification, link prediction tasks, and qualitative analysis.

For future work, a possible direction is to extend GAT2VEC to learn representations of vertices in a heterogeneous network leveraging diverse informations associated with different types of vertices.

References

1. Blei DM, Ng AY, Jordan MI (2003) Latent dirichlet allocation. J Mach Learn Res 3:993–1022. https://doi.org/10.1162/jmlr.2003.3.4-5.993
2. BLOGCATALOG (2017). http://dmml.asu.edu/users/xufei/datasets.html. Accessed 01 July 2017
3. Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: a survey. ACM Comput Surv 41(3):15:1–15:58. https://doi.org/10.1145/1541880.1541882
4. CITESEER (2017). http://citeseerx.ist.psu.edu/. Accessed: 01 July 2017
5. DBLP (2017). http://arnetminer.org/citation. Accessed: 01 July 2017
6. Dong Y, Chawla NV, Swami A (2017) Metapath2vec: scalable representation learning for heterogeneous networks. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge

discovery and data mining, KDD'17. ACM, New York, pp 135–144. https://doi.org/10.1145/3097983.3098036

7. Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ (2008) LIBLINEAR: a library for large linear classification. J Mach Learn Res 9:1871–1874. https://doi.org/10.1145/1390681.1442794

8. Fouss F, Pirotte A, Renders JM, Saerens M (2007) Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. IEEE Trans Knowl Data Eng 19(3):355–369. https://doi.org/10.1109/TKDE.2007.46

9. Grover A, Leskovec J (2016) Node2vec: scalable feature learning for networks. In: Proceedings of the 22Nd ACM SIGKDD international conference on knowledge discovery and data mining, KDD'16. ACM, New York, pp 855–864. https://doi.org/10.1145/2939672.2939754

10. Hamilton WL, Ying R, Leskovec J (2017) Inductive representation learning on large graphs. CoRR. arXiv:1706.02216

11. Huang X, Li J, Hu X (2017) Label informed attributed network embedding. In: Proceedings of the 10th ACM international conference on web search and data mining, WSDM'17. ACM, New York, pp 731–739. https://doi.org/10.1145/3018661.3018667

12. Kefato ZT, Sheikh N, Montresor A (2017) Mineral: multi-modal network representation learning. In: Machine learning, optimization, and big data—3rd international conference, MOD 2017, Volterra, Italy, September 14–17, 2017, Revised Selected Papers, pp 286–298. https://doi.org/10.1007/978-3-319-72926-8_24

13. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. CoRR. arXiv:1609.02907

14. Le QV, Mikolov T (2014) Distributed representations of sentences and documents. CoRR. arXiv:1405.4053

15. Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. J Am Soc Inf Sci Technol 58(7):1019–1031. https://doi.org/10.1002/asi.v58:7

16. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. CoRR. arXiv:1301.3781

17. Morin F, Bengio Y (2005) Hierarchical probabilistic neural network language model. In: Proceedings of the 10th international workshop on artificial intelligence and statistics, AISTATS'05. Society for Artificial Intelligence and Statistics, Bridgetown, Barbados

18. Pan S, Wu J, Zhu X, Zhang C, Wang Y (2016) Tri-party deep network representation. In: Proceedings of the 25th international joint conference on artificial intelligence, IJCAI'16. AAAI Press, New York, pp 1895–1901. http://dl.acm.org/citation.cfm?id=3060832.3060886

19. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, KDD'14. ACM, New York, pp 701–710. https://doi.org/10.1145/2623330.2623732

20. Sen P, Namata GM, Bilgic M, Getoor L, Gallagher B, Eliassi-Rad T (2008) Collective classification in network data. AI Mag 29(3):93–106. http://www.cs.iit.edu/~ml/pdfs/sen-aimag08.pdf. Accessed June 2017

21. Shaw B, Jebara T (2009) Structure preserving embedding. In: Proceedings of the 26th annual international conference on machine learning, ICML'09. ACM, New York, pp 937–944. https://doi.org/10.1145/1553374.1553494

22. Tang J, Liu J, Zhang M, Mei Q (2016) Visualizing large-scale and high-dimensional data. In: Proceedings of the 25th international conference on world wide web, WWW'16. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Geneva, pp 287–297. https://doi.org/10.1145/2872427.2883041

23. Tang J, Qu M, Mei Q (2015) Pte: predictive text embedding through large-scale heterogeneous text networks. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, KDD'15. ACM, New York, pp 1165–1174. https://doi.org/10.1145/2783258.2783307

24. Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q (2015) Line: large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web, WWW'15. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Geneva, pp 1067–1077. https://doi.org/10.1145/2736277.2741093

25. Tang L, Liu H (2009) Relational learning via latent social dimensions. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining, KDD'09. ACM, New York, pp 817–826. https://doi.org/10.1145/1557019.1557109

26. Tang L, Liu H (2011) Leveraging social media networks for classification. Data Min Knowl Disc 23(3):447–478. https://doi.org/10.1007/s10618-010-0210-x
27. Wang D, Cui P, Zhu W (2016) Structural deep network embedding. In: Proceedings of the 22Nd ACM SIGKDD international conference on knowledge discovery and data mining, KDD'16. ACM, New York, pp 1225–1234. https://doi.org/10.1145/2939672.2939753
28. Wang X, Tang L, Gao H, Liu H (2010) Discovering overlapping groups in social media. In: Proceedings of the 10th IEEE international conference on data mining, ICDM 2010. IEEE, Sydney, pp 569–578. https://doi.org/10.1109/ICDM.2010.48
29. Yang C, Liu Z, Zhao D, Sun M, Chang EY (2015) Network representation learning with rich text information. In: Proceedings of the 24th international conference on artificial intelligence, IJCAI'15. AAAI Press, New York, pp 2111–2117. http://dl.acm.org/citation.cfm?id=2832415.2832542
30. Yang Z, Cohen WW, Salakhutdinov R. (2016) Revisiting semi-supervised learning with graph embeddings. CoRR. arXiv:1603.08861
31. Yu X, Ren X, Sun Y, Gu Q, Sturt B, Khandelwal U, Norick B, Han J (2014) Personalized entity recommendation: a heterogeneous information network approach. In: Proceedings of the 7th ACM international conference on web search and data mining, WSDM'14. ACM, New York, pp 283–292. https://doi.org/10.1145/2556195.2556259