24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

A Multilayered Informative Random Walk for Attributed Social Network Embedding

Sambaran Bandyopadhyay[1] and **Anirban Biswas**[2] and **Harsh Kara**[3] and **M. N. Murty**[4]

Abstract. Network representation learning (also known as Graph embedding) is a technique to map the nodes of a network to a lower dimensional vector space. Random walk based representation techniques are found to be efficient as they can easily preserve different orders of proximities between the nodes in the embedding space. Most of the social networks now-a-days have some content (or attributes) associated with each node. These attributes can provide complementary information along with the link structure of the network. But in a real life network, the information carried by the link structure and that by the attributes vary significantly over the nodes. Most of the existing unsupervised attributed network embedding algorithms do not distinguish between the link structure and the attributes of a node depending on their informativeness.

In this work, we propose an unsupervised node embedding technique that exploits both the structure and attributes by intelligently prioritizing one of them, in the random walk, for each node separately. We convert the network into a multi-layered graph and propose a novel random walk based on the informativeness of a node in different layers. This unified approach is simple and computationally fast, yet able to use the content as a complement to structure and vice-versa. Experimental evaluations on four real world publicly available datasets show the merit of our approach (up to 168.75% improvement) compared to the state-of-the-art algorithms in the domain. We make the source code available to download.

1 INTRODUCTION

Social networks are ubiquitous in our daily life. A network is typically represented using a graph. The success of various mining tasks depends on efficient feature selection and design of the network. But efficient feature engineering needs a lot of domain knowledge and human efforts, especially when the networks are large and sparse as in the real-world applications. Compared to that, in a graph embedding framework [20], a function to represent each node in the form of a compact and dense vector is learnt mostly in a task independent way. As shown in the literature [26, 28], machine learning algorithms perform better on these embeddings for network mining tasks such as node classification, community detection, etc. Recently, embedding based approaches shows promising results for other tasks in social networks like classify crisis related tweets [1] and identification of abusive texts [6].

One important goal of the graph embedding algorithms is to preserve different orders of node proximities into the embedding space, i.e., nodes which belong to a local neighborhood of each other in the graph should also be close in the vector space. A random walk on a graph produces a sequence of nodes such that the nodes within a small window of this sequence belong to a local neighborhood of each other. Naturally, random walk has been a fundamental backbone to many popular graph embedding algorithms as shown in [9, 22]. But most of these random walk based techniques only consider the link structure of the network. However, real world social networks have rich attributes (or content) associated with each node. A network where each node has a set of attribute values is called an attributed network or attributed graph. Node attributes can complement the structural information, specially when the structure is noisy. For example, there are millions of active and connected users in Twitter and each of them can post thousands of tweets. Content of these tweets can include text, images or videos. Naturally the connections between users in such social networks also depend on the content that they post. Sociological theories such as homophily [17] also suggest a strong correlation between the structure and the content of a network.

Node attributes have been shown to be useful for some network mining tasks such as tackling filter bubble problem [15], evolving social action prediction [25], etc. Combining node attributes with the link structure has also improved the quality of node embeddings in the networks. However, most of the existing attributed network embedding techniques are either matrix factorization based techniques [34, 13] or graph neural network based techniques [14, 28]. Matrix factorization based techniques, being linear in nature, often fail to address the highly non-linear characteristics of a network. Graph neural network techniques [14], though proved to be efficient in the literature, are mostly semi-supervised in nature as they need labels of the nodes to learn large number of parameters of the graph neural networks. This is a serious restriction for real world social networks where node labels (for e.g., the community membership of a node) are generally expensive to obtain. Further, a recent work [35] on graph neural networks shows that following are the two important properties for any node embedding algorithm in general:

- A node embedding is **position aware** if there exists a function which can map the distances between the embedding of any two nodes into their corresponding shortest path distance in the network.
- A node embedding is **structure aware** if it is a function of up to q-hop network neighbourhood (for some positive integer q) of a node in the network.

It has been shown that graph neural networks are structure aware, as they aggregate attributes from the neighborhood, but they may not be position aware [35]. Random walk based techniques can satisfy

[1] IBM Research & IISc, Bangalore, email: samb.bandyo@gmail.com
[2] Indian Institute of Science, Bangalore, email: anirbanb@iisc.ac.in
[3] Indian Institute of Science, Bangalore, email: erharshkara@gmail.com
[4] Indian Institute of Science, Bangalore, email: mnm@iisc.ac.in

both the properties as: (i) A single random walk from a node can preserve the graph distances in the embedding space of any pair of nodes which are within a context window (explained in Section 4.1) length away and use of negative sampling push the embeddings of any two randomly sampled nodes far from each other. (ii) A random walk from a node cover multiple nodes (can cover all the nodes if the number of random walks are more) from some q-hop neighborhood, making the node embeddings structure aware. With this motivation and addressing the above research gaps, we propose a random walk based unsupervised approach which uses link structure and attributes of the nodes as complement to each other to generate node embeddings of a graph.

Integrating attributes of each node into the state-of-the-art unsupervised graph embedding techniques is challenging for real world networks. Often the structure and the attributes of a node are not consistent with each other. There can be significant gap in the semantic information carried by them. Figure 1 explains an example where for a particular node (colored in blue), content is *more informative* than its structure. It is important to exploit the extent of informativeness of an individual node in its structure and attribute layers for network embedding. However, existing unsupervised attributed network embedding techniques [13, 8] process link structure and the node attributes uniformly. Semi-supervised graph embedding algorithms such as [28] which employs attention mechanism to characterize the importance of a node, are difficult to apply when node labels are too expensive to get. We also address this research gap in this paper. Following are the **contributions** we make:

- We propose a novel unsupervised algorithm **MIRand** (Multilayered Informative Random Walk), which creates a multi-layer graph and employs a novel random walk that exploits the *informativeness* of a node by unifying its structure and attributes. To the best of our knowledge, this is the first attributed graph embedding technique which employs a multi-layered graph random walk where one layer corresponds to structure and the other deals with the attributes associated with the nodes. Thus, our algorithm can be seen as a novel extension of random walk based network embedding approaches (such as node2vec [9]) for attributed network.
- We evaluate the performance of the proposed algorithm on real world datasets on multiple downstream machine learning tasks. We are able to improve the performance by **168.75%** of the best of the baselines for some tasks. Source code of MIRand is publicly available at `https://github.com/anirban-code-to-live/mirand` to reproduce the results.

2 RELATED WORK

We refer to the literature survey in [11, 32] for a comprehensive overview on network embedding. However for the sake of completeness, we discuss some of the more prominent works in this section. Standard dimensionality reduction techniques or hand crafted feature engineering techniques [7] were popular for graph mining tasks for a long time. Development in natural language processing domain [18] influenced the field of network representation learning. DeepWalk [20] and node2vec [9] employ random walks to build a corpus of node sequences and generate the node embeddings by maximizing the log likelihood of the context of each node from the corpus. Line [26] uses two different optimization formulations to explicitly capture the first order and second order proximities in node embeddings. Autoencoder based network embedding approaches are proposed in

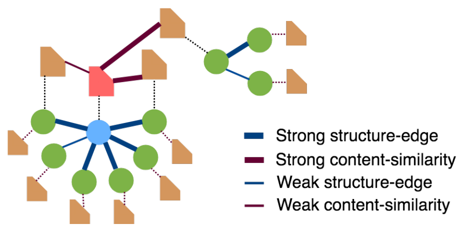

Figure 1: It is a sample network where each node (circle) has some content (connected by dotted lines) associated with it. If we consider the central node in blue, it is strongly connected (with high edge weights) to many other nodes in the network. So it is likely to be less similar to any of them [16]. But if we consider the same node in the content layer, it is strongly similar to only the content of few other nodes in the network. Hence content is semantically more informative for this node.

[30, 5]. Struc2vec [22] is another random walk based node embedding strategy which finds similar embeddings for the nodes which are structurally similar. Generative adversarial learning framework has also been explored for graph representation in [31]. Theoretical analysis to show the correspondence between random walk based embedding approaches and matrix factorization is done in [21]. All of these methods consider only the link structure of the network.

TADW [34] is the first approach to propose a matrix factorization approach to combine link structure and content for network embedding. Joint matrix factorization techniques to obtain unsupervised node embeddings for attributed networks are also present in the literature [13, 3]. A deep architecture consisting of two parallel autoencoders for structure and attributes is proposed in [8]. Another deep autoencoder based approach to use the relation between structure and content for node embedding is proposed in [12]. A matrix factorization based approach for outlier aware network embedding is developed in [4]. [37] proposes an embedding method for textual content in a network, along with the embedding of the nodes, by employing a network diffusion process. These approaches are unsupervised, but they don't give different importance to structure and content based on their role in the network.

Semi-supervised graph neural network (GNN) [19] based approaches are also proposed in the recent literature for attributed graph embedding. A semi-supervised graph convolution network (GCN) model to aggregate information from the neighbor nodes is proposed in [14]. [10] proposes GraphSAGE, which is an extension of GCN with different types of information aggregation with neighborhood sub-sampling. Attention based graph embedding is introduced in [28] and further extended to multiple attention heads with different importance in [36]. These algorithms being semi-supervised in nature, cannot be applied to cases where no labeling information is available. In contrast to them, we propose a novel unsupervised random walk based embedding approach which uses the informativeness of a node in terms of its structure and content.

3 PROBLEM STATEMENT

An information network is typically represented by a graph $G = (V, E, W, F)$, where $V = \{v_1, v_2, \cdots, v_n\}$ is the set of nodes (a.k.a. vertexes), each representing a data object. $E \subseteq \{(v_i, v_j) | v_i, v_j \in V\}$ is the set of edges between the vertexes. Each edge $e \in E$ is an ordered pair $e = (v_i, v_j)$ and is associated with a weight $w_{v_i,v_j} > 0$, which indicates the strength of the relation. W is the set of all those weights. If G is undirected, we have $(v_i, v_j) \equiv (v_j, v_i)$ and $w_{v_i,v_j} = w_{v_j,v_i}$; if G is unweighted, $w_{v_i,v_j} = 1, \forall (v_i, v_j) \in E$. $F = \{f_i \mid i \in \{1, 2, \cdots, n\}\}$, where $f_i \in \mathbb{R}^d$ is the attribute vector (or content) associated with node $v_i \in V$. So F can be considered as the content matrix. If content is textual in nature, F can be represented by bag-of-word model where typically stop words are removed and stemming is done as preprocessing steps. Each row of this matrix is a tf-idf vector for the textual content at the corresponding node. The dimension of F is $n \times d$, where d is the number of unique words (attributes) after preprocessing the corpus. This can be generalized easily to other types of content such as image, video, or speech.

Given G, the task is to find some low dimensional vector representation of the nodes of G which is consistent with both the structure of the network and the content of the nodes. More formally, for the given network G, the network embedding is to learn a function $f : v_i \mapsto \mathbf{x}_i \in \mathbb{R}^K$, i.e., it maps every vertex to a K dimensional vector, where $K < min(n, d)$. The representations should preserve the underlying semantics of the network. Hence the nodes which are close to each other in terms of their positional distance or similarity in content should have similar representation. This representation should also be compact and continuous as that would help the conventional machine learning algorithms to perform better on downstream network mining tasks.

4 SOLUTION APPROACH: MIRand

This section describes the details of our proposed approach MIRand. Given the network G with content in the form of a matrix F, our first goal is to divide the network into two layers. The first layer corresponds to the structure, and the second layer is for the content.

Structure Layer: Intuitively, this layer is the same as the given network without any content (or attribute) in the nodes. Mathematically, given the input network $G = (V, E, W, F)$ (as in Section 3), the structure layer is a graph $G_s = (V_s, E_s, W_s)$, with $E_s = E$ and $W_s = W$. The node set is exactly same as the given network, only they are denoted with a superscript ‘s’. So $v \in V \iff v^s \in V_s$ for all nodes $v \in V$.

Content Layer: This layer is a directed graph which captures the similarity between pairs of nodes in terms of their respective contents or attributes. Again, given the input network G as above, we define the content layer to be the graph $G_c = (V_c, E_c, W_c)$. We denote the nodes with the superscript ‘c’ added, $v \in V \iff v^c \in V_c$, $\forall v \in V$ in the content layer. For each node $v_i \in V$, one can potentially compute the similarity or weight to all other nodes v_j $(j \neq i)$ based on the cosine similarity of the row vectors $F_{i.}$ and $F_{j.}$. So, $w^c_{ij} = Cosine(F_{i.}, F_{j.})$. But in this case, the content graph would be nearly complete (i.e., there is an edge between almost any pair of vertices) and it would increase the computational time to process the graph. Whereas, in the structure layer, the number of edges is fixed. So we first compute average number of outgoing edges over all the nodes in the structure layer. Let’s call it avg_s. If the given network is directed, $avg_s = |E|/n$, and if it is undirected then, $avg_s = 2 \times |E|/n$. We want the number of edges in the content layer to be comparable with the number of edges in the structure layer, so that it can help the random walk as discussed in Section 4.1. So in the content layer, for each node v^c_i, we find the top $\theta \times \lceil avg_s \rceil$ nodes (excluding v^c_i) in terms of their content cosine similarities with v^c_i, where θ is called ratio parameter, $\theta \in \mathbb{R}_+$. Then we add a directed edge from v^c_i to each of those nodes with the edge weight as the cosine similarity between the respective contents[1]. We also discard the edges which have negative weights, i.e., cosine similarity of the two end vertices is negative. As explained in Section 4.1, θ controls the time the random walk may spend in a layer. This has also been shown in the experiments. Figure 2 shows both structure and content layers, and their interconnections which we will discuss next.

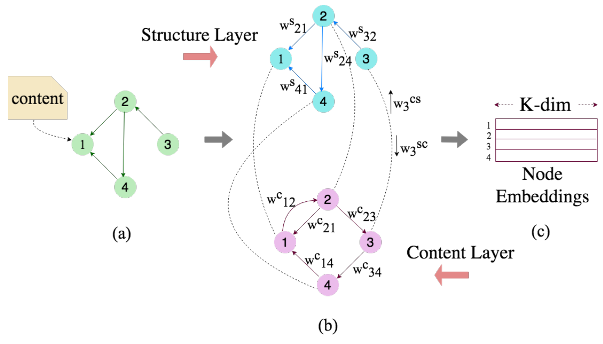

Figure 2: This demonstrates the work flow of MIRand. (a) Represents an information network with each node having some content (attributes) associated with it. (b) Represents the two-layered network. The upper part depicts the structure layer which can be (un)directed and (un)weighted depending on the input network. The lower part is for the content portion which is directed and weighted network formed using the method described. Inter layer (weighted and directed) edges connecting the corresponding nodes are shown by dashed lines. (c) Shows the K dim. output embeddings.

4.1 Random Walk based on Informativeness of a node

In this subsection, we propose a novel random walk which would traverse the nodes in structure and content layers intelligently. So we first create the structure layer and the content layer as discussed before. Then we connect nodes such that for any node v_i in the original network, suppose the label of it in structure network is v^s_i and the label in content layer is v^c_i. We add two directed and weighted edges between v^s_i and v^c_i as explained later. At any point of time when the random walk reaches node v_i, our goal is to select the structure or the content layer, based on the *informativeness* of node v_i in the corresponding layer. We explain the notion of informativeness of a node below.

For any random walk to capture the inherent homophily property of an attributed network, it is important to concentrate on the nodes which are *strongly connected* with only a few neighbors, so that next step of a random walk from the node is more informative. On the contrary, a node which is strongly connected to a large number of nodes in an information network is actually less similar to most of its neighbors [16, 22]. In this work, we assume a strong (edge) connection is the one whose edge weight is more than the average edge weight in the graph layer. So we define Γ^s_i as the set of neighbors strongly connected to v^s_i in the structure layer.

$$\Gamma^s_i = \{v^s_j \in V_s \mid w^s_{(v^s_i, v^s_j)} \geq \frac{1}{|E_s|} \sum_{e' \in E_s} w^s_{e'}\} \tag{1}$$

So each node in Γ^s_i has an incoming edge from v^s_i whose weight is

[1] That’s why the content layer is always directed and weighted.

more than the average edge weight of the structure layer[1]. $|\Gamma_i^s|$ is the number of strongly connected neighbors of v_i in the structure layer. Similarly, Γ_i^c for the content layer is defined as:

$$\Gamma_i^c = \{v_j^c \in V_c \mid w^c_{(v_i^c, v_j^c)} \geq \frac{1}{|E_c|} \sum_{e' \in E_c} w^c_{e'}\} \tag{2}$$

We define the *informativeness* of nodes v_i^s and v_i^c respectively for a random walk as:

$$I_i^s = \frac{1}{\ln(e + |\Gamma_i^s|)}, \quad I_i^c = \frac{1}{\ln(e + |\Gamma_i^c|)} \tag{3}$$

Now, let us consider the directed edge (v_i^s, v_i^c) with weight w_i^{sc}. As discussed, at any point of time the random walk selects either the structure or the content layer depending on the informativeness of v_i^s and v_i^c, given that the present node is v_i. The probability of selecting structure layer is proportional to w_i^{cs} and that of selecting content layer is proportional to w_i^{sc}. Hence we set these inter layer weights as: $w_i^{sc} = I_i^c$ and $w_i^{cs} = I_i^s$, $\forall v_i \in V$. The complete multi-layered network is shown in Figure 2. Next we define the inter-layer and intra-layer transition probabilities of the biased random walk on this multi-layered graph.

The random walk proceeds as follows. Given, at a particular time-step of the random walk, we are at node v_i, either in the structure or in the content layer. Before taking the next step, we first calculate the probability of taking that step either into the structure layer or into the content layer. Our goal is to move to a layer where node v_i is more informative. So we define the inter-layer transition probabilities as:

$$p(v_i^s|v_i) = \frac{w_i^{cs}}{w_i^{sc} + w_i^{cs}} = \frac{I_i^s}{I_i^s + I_i^c} \tag{4}$$

$$p(v_i^c|v_i) = 1 - p(v_i^s|v_i) = \frac{w_i^{sc}}{w_i^{sc} + w_i^{cs}} = \frac{I_i^c}{I_i^s + I_i^c} \tag{5}$$

In the context of the random walk, smaller the value of I_i^s, larger the number of edges with large edge weight from node v_i^s. So the random walk has many choices to move from node v_i^s if it remains to be in the structure layer. Whereas, if the value of I_i^c is high, then there are less number of outgoing edges with large edge weight from the node v_i^c. In this case for the random walk at node v_i, the choice is more informative and less random in the content layer than that in the structure layer. Hence when the value of I_i^c is high, we want to prefer content layer, and similarly when the value of I_i^s is high, we want to prefer structure layer.

Now we discuss the probability of selecting the next vertex from the current vertex v_i, given that we have selected a particular layer, as discussed above. So we define the intra-layer transition probabilities as follows. Suppose the second order random walk was at node $v_i^{l_1}$ at time $t-1$ and it is at node $v_j^{l_2}$ at time t, where l_1 and l_2 denote the corresponding layers, $l_1, l_2 \in \{s, c\}$. Here we extend the (unnormalized) transition probabilities from [9] of selecting a node $v_k^{l_2}$ in layer l_2 (as currently it is at l_2 layer at time t) as:

$$P(v_k^{l_2}|v_j^{l_2}, v_i^{l_1}) = \begin{cases} w_{jk}^{l_2}, & \text{if } e_{ij}^{l_2} \notin E_{l_2} \\ \frac{1}{p} w_{jk}^{l_2}, & \text{if } e_{ij}^{l_2} \in E_{l_2} \text{and } d_{ik}^{l_2} = 0 \\ w_{jk}^{l_2}, & \text{if } e_{ij}^{l_2} \in E_{l_2} \text{and } d_{ik}^{l_2} = 1 \\ \frac{1}{q} w_{jk}^{l_2}, & \text{if } e_{ij}^{l_2} \in E_{l_2} \text{and } d_{ik}^{l_2} = 2 \end{cases} \tag{6}$$

[1] For an unweighted network, all the outgoing neighbors of a node are strongly connected to that node.

The first case considers the scenario when there is a change of layer (i.e., $l_1 \neq l_2$) at time t and the node v_i is no longer directly connected to the node v_j in the new layer l_2. In this case the random walk just selects the next node based on the weights of the outgoing edges from the node v_j in layer l_2. It is also to be noted that if $l_1 = l_2$ then $e_{ij}^{l_2} \in E_{l_2}$, but the reverse is not necessarily true. $d_{ik}^{l_2}$ is the unweighted distance between the nodes $v_i^{l_2}$ and $v_k^{l_2}$. There are two hyper parameters $p, q > 0$. A larger value of p would force the random walk not to visit the node visited last time. Similarly, a larger q would force the random walk not to move away beyond 1 hop neighborhood of the recently visited node. Thus they control the search to follow BFS or DFS strategy accordingly and can be fixed as shown in [9]. The next node selected by the process is added to the sequence of the random walk, after discarding the layer information s or c. This ensures that we get only one embedding for each node, unlike as in [8] which needs to concatenate embeddings for structure and attributes as a post processing step. We repeat the above step l times, l being the length of each truncated random walk from a node.

Algorithm 1 MIRand - Multilayered Informative Random Walk for Graph Embedding with Content

```
    Input: The network $G = (V, E, W, F)$, $K$: Dimension of the embedding space where $K << min(n, d)$, $\theta$: ratio parameter to construct the content layer, $r$: Number of times to start random walk from each vertex, $l$: Length of each random walk
    Output: The node embeddings of network $G$
 1: Generate the structure layer and the content layer (using the hyper-parameter $\theta$) and complete the multilayered network by setting the inter-layer weights.
 2: $Corpus$ = [ ]                                   ▷ Initialize the corpus
 3: for $iter \in \{1, 2, \cdots, r\}$ do
 4:     for $v \in V$ do
 5:         $Walk$ = [$v$]    ▷ Initialize the Walk (sequence of nodes) for the truncated random walk
 6:         for $walkIter \in \{1, 2, \cdots, l\}$ do
 7:             Select the layer (w.r.t. the last appended node) with inter layer transition probabilities as in Eq. 4 and 5 to move next
 8:             Find the next node $v_i$ within the selected layer by using intra layer transition probabilities (Eq. 6)
 9:             Append $v_i$ to $Walk$
10:         end for
11:         Append Walk to Corpus
12:     end for
13: end for
14: Find the node embeddings by optimizing Eq. 7
```

Once the desired number of truncated random walk sequences are generated for each node, we maximize the log probability of the context (nodes which appear within a specified left and right window) of a node in a random walk sequence with respect to that node [20]. More formally, we want to maximize the following:

$$\max_X \sum_{v \in V} \log P(C(v)|\mathbf{x}_v) = \max_X \sum_{v \in V} \sum_{u \in C(v)} \log P(u|\mathbf{x}_v)$$

Here $\mathbf{x}_v \in \mathbb{R}^K$ is the embedding of node v and X is the set of embeddings of all the nodes. $C(v)$ is the set of nodes in the context of the node v in a random walk. We also assume conditional independence of the nodes in a context. Each of the above probabilities can be represented using standard softmax function parameterized by the dot product of $\mathbf{x}_u$ and $\mathbf{x}_v$. Finally the above equation amounts to the following:

$$\max_{X} \sum_{v \in V} \sum_{u \in C(v)} \big(\mathbf{x}_u \cdot \mathbf{x}_v - \log Z_v \big) \tag{7}$$

$Z_v = \sum_{u' \in V} \exp(\mathbf{x}_{u'} \cdot \mathbf{x}_v)$ is the partition function, which can be approximated by negative sampling [18]. The optimization in Eq. 7 can be solved using stochastic gradient ascent over the model parameters. Algorithm 1 summarizes the whole process of MIRand. The two layered network that we have created is a special type of heterogeneous network. But our approach is inherently different from any heterogeneous network embedding such as [24], as MIRand exploits the one to one correspondence between structural and content views of the same node and extract the complementary information.

Please note that, MIRand can easily be extended for social networks which have more than one type of content (for e.g., users in Facebook where they post texts, images and videos). In that case, instead of two layers, we can have multiple layers (one for structure and then one layer for each type of content) and a random walk traversing all of them using intra and inter-layer transition probabilities.

4.2 Time and Space Complexity of MIRand

If we assume that the number of random walks from each node and the length of each random walk are constant, then the time complexity of the random walk of MIRand is $O((|V| + |E|) \log |V|)$, due to the use of alias table, as shown done in [9]. The only overhead for MIRand is the construction of the content network, which in the worst case can be $O(|V|^2)$. But as the content layer is also sparse in nature, the construction of this layer can be completed in $O(|V|log|V|)$ time by using Space Partitioning Tree [23] that allows us to find the closest object to another object in logarithmic time. So the total time complexity of MIRand is $O(|V|log|V| + (|V| + |E|) \log |V|) = O((|V| + |E|) \log |V|)$.

The structure layer of MIRand is same as the given network. Content layer has θ times the number of edges of the structure layer, where θ is a constant. Hence MIRand needs only $O(|V| + |E|)$ space to store the graph in the edge list format. Interestingly, this analysis shows that, though MIRand is able to use both structure and the attributes of a network, its asymptotic time and space complexity are the same as the standard random walk based algorithm such as node2vec, which use only structure of the network.

Table 1: Summary of the datasets used: Each dataset has both network structure and textual content, with labeled communities. *#Attributes* is the length of the attribute vector associated with each node. *Inter/intra* is the ratio of the number of inter community links to that of the intra community links.

Dataset	#Nodes	#Edges	#Labels	#Attributes	Inter/Intra
Cora	2708	5429	7	1433	0.22
Citeseer	3312	4715	6	3703	0.34
Pubmed	19717	44338	3	500	0.25
Flickr	7575	239738	9	12047	3.19

5 EXPERIMENTAL EVALUATION

In this section, we experimentally evaluate the performance of the proposed algorithms and compare the results with state-of-the-art network embedding algorithms. To evaluate the quality of the generated embeddings, we have selected three different network mining tasks: network visualization, node classification and node clustering.

5.1 Datasets Used

We have used 4 publicly available real world network datasets with ground truth community labels in this paper. A detailed description is provided in Table 1. **Cora**[5], **Citeseer**[5] and **Pubmed**[5] are the citation networks, where citations between the papers form the network. Each node also has some attribute coming from the meta data and content of the corresponding scientific paper. **Flickr**[6] is an online social network where people can share photographs and can also follow each other which forms the links of the network. The tags specified on the image act as attributes. The group which the photographer has joined acts as its label.

5.2 Baseline Algorithms and Experimental Setup

Recently, a number of network embedding algorithms are proposed in the literature. In this paper, we propose an unsupervised attributed network embedding algorithm. To make a fair comparison, we have selected only the following state-of-the-art node embedding algorithms which are **unsupervised** in nature and have **publicly available source code**.

- **DeepWalk** [20] and **node2vec** [9] are random walk based node embedding techniques. DeepWalk uses a first order random walk whereas node2vec uses 2nd order biased random walk.
- **SDNE** [30] is a deep autoencoder based node embedding technique which enforces homophily constraint explicitly.
- **Struc2Vec** [22] is a random walk based technique which finds similar embeddings for the nodes which are structurally similar.
- **TADW** [34] and **AANE** [13] are matrix factorization based technique which uses both link structure and node attributes.
- **GraphSAGE** [10] is a deep graph convolution neural network based technique which uses neighborhood sub-sampling and different neighborhood node attribute aggregation. We use the unsupervised version of GraphSAGE in a non-inductive setting, i.e., all the nodes in the network were present from the beginning.
- **DGI** [29] is a recent graph convolution network based unsupervised approach by maximizing mutual information between patch representations and corresponding high-level summaries of a graph, obtained by using both link structure and node attributes.

Please note some other popular network embedding algorithms such as GCN [14] and GAT [28] are not used as baselines as they are semi-supervised in nature. We have kept the embedding dimension (K) for all the algorithms and on all the datasets to be 128. For MIRand, the value of θ to be 2 and the values of p and q (in Eq. 6) are 1 each. The effect of changing these hyper-parameters on the performance of MIRand is also discussed in Section 5.6.

5.3 Network Visualization

The first downstream machine learning task that we consider is network visualization. Here, the goal is to visually separate the nodes which belong to different communities in the network. First, the 128

[5] `https://linqs.soe.ucsc.edu/data`
[6] `https://github.com/xhuang31/AANE_Python/blob/master/Flickr.mat`

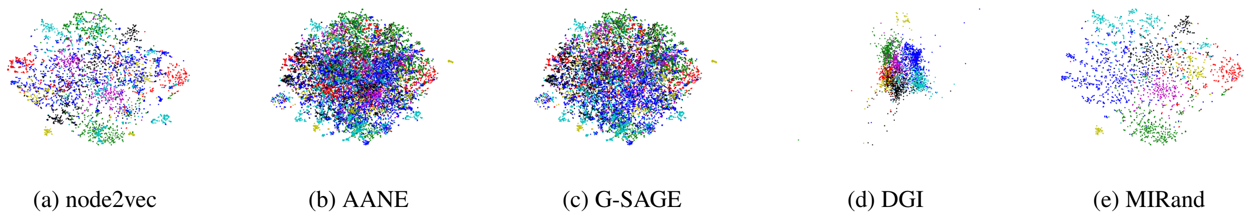
(a) node2vec (b) AANE (c) G-SAGE (d) DGI (e) MIRand

Figure 3: t-SNE Node Visualization on Cora. Different colors represent different communities (G-SAGE ≡ GraphSAGE).

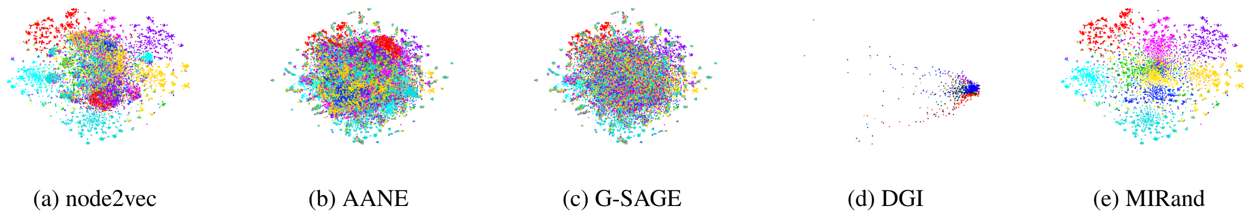
(a) node2vec (b) AANE (c) G-SAGE (d) DGI (e) MIRand

Figure 4: t-SNE Node Visualization on Flickr dataset. Different colors represent different node communities.

dimensional embeddings of the nodes are given as input to t-SNE [27], which maps them to a 2 dimensional space. Then we plot these 2 dimensional points (nodes) in Figures 3 and 4 for Cora and Flickr datasets respectively. Different colors are used to label nodes from different communities. Due to page limitation, we have shown the performance of only a subset of better performing baselines for this task. As the figures show, the performance of MIRand is clearly better than all the baseline, over all the four datasets we used. MIRand is able to distinguish communities (having different colors) well compared to other algorithms. Among the baselines, node2vec, another unsupervised random walk based approach, is able to perform good. The better empirical performance of MIRand indicates that it is able to intelligently exploit the structure and content of the nodes compared to all the baselines. It is also to be noted that approaches like AANE, GraphSAGE and DGI, though use node attribute along with link structure of the networks, do not perform well for visualization.

5.4 Multi-class Node Classification

Node classification is a useful network mining task in cases when labeling information is available only for a small subset of nodes in the network. For this task, we train a logistic regression classifier on the node embeddings as features. We split the set of nodes of the graph into training set and testing set. The training set size is varied from 10% to 50% of the entire data. The remaining (test) data is used to compare the performance of different algorithms. We choose Macro-F1 and Micro-F1 to measure the performance of multi-class classification algorithms. Normally, higher the values are, better is the classification performance. We repeat each experiment 10 times and report the average results.

The classification results are presented in Figure 5. First, we observe that, except for the dataset Cora, MIRand is able to outperform all the baseline algorithms. The performance margin is prominent on Citeseer and Flickr for all the training sizes. This margin is often larger for smaller training sizes, which is more desirable as collecting ground truth label is expensive in networks. For Pubmed, the performances of DeepWalk, node2vec and AANE are very close to MIRand. On Cora dataset, DeepWalk and node2vec are able to outperform MIRand in most of the cases. Interestingly, on Flickr dataset where the total number of edges between the communities is more than that within the communities (Tab. 1), MIRand is consistently able to exceed 80% accuracy for node classification because of the efficient use of the content layer. Most of the baselines fail to classify the nodes well because of the poor link structure of this dataset.

5.5 Community Detection

Community detection or node clustering is an unsupervised method of grouping the nodes into multiple communities. We use KMeans++ [2] to cluster the node embeddings produced by different algorithms. We use unsupervised accuracy [33, 4], which is a metric to measure the quality of (unsupervised) clustering against the ground truth labels of the objects (nodes in this case). Intuitively, it uses different permutations of the labels and chooses the label ordering which gives best possible accuracy. The performance is shown in Figure 6. One can see that the MIRand outperforms all the state-of-the-art algorithms, except on Pubmed, where DeepWalk and node2vec perform marginally better than it. On Flickr dataset, none of the baseline algorithms performs good, but MIRand is still able to exploit structure and content as complementing each other and deliver a good accuracy. We see a performance gain of **168.75%** by MIRand compared to node2vec which is the second best performing algorithm on this dataset. Superior performance of MIRand on Flickr dataset can be observed in node classification also. Flickr, being a dense dataset and having more number of inter-community edges than that of intra-community, simple random walk based techniques fail.

It is important to note that for clustering on the Pubmed dataset, only structure based methods (DeepWalk and node2vec) perform better than others, except MIRand again. It means MIRand is able to switch intelligently between the structure and content layers during the learning process and embeddings were learnt mostly from the structure layer. However, struc2vec performs poorly on most of the datasets as this algorithm learns similar embedding of the nodes which are just structurally similar to each other. Thus MIRand is a robust algorithm and less prone to such inconsistencies and noise.

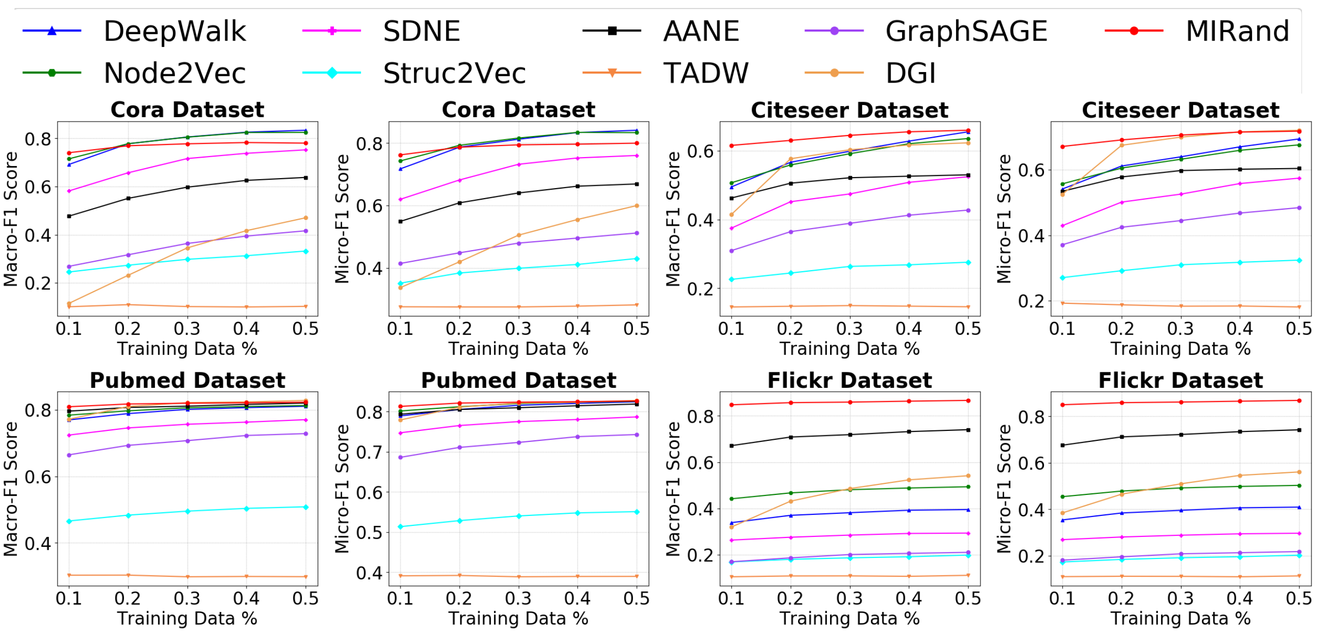

Figure 5: Performance of different embedding algorithms for Classification with Logistic Regression

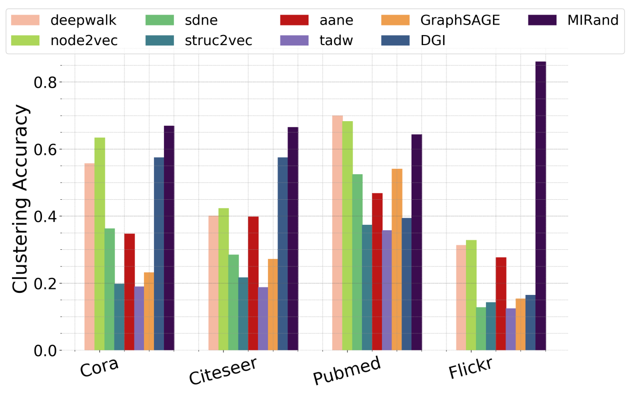

Figure 6: Community detection by embedding algorithms

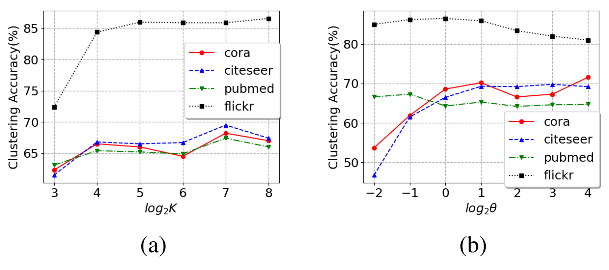

Figure 7: Community Detection performance (measured by unsupervised clustering accuracy) of MIRand with (a) different values of embedding subspace dimension K, (b) different values of ratio parameter θ.

5.6 Parameter Sensitivity and Analysis

In this section, we vary two important parameters, dimension of the embedding space K and the ratio parameter θ of MIRand algorithm. We choose node clustering as the downstream machine learning task to show the performance on varying these parameters. As expected, when the dimension of the embedding subspace is very less (in Fig. 7.a), it becomes difficult to capture all the inherent features of the attributed nodes. As a result, performance deteriorates. We observed optimal performance when $K = 128$ for most of the datasets, after which the performance degrades because of the inclusion of redundant or noisy features.

From Fig. 7.b, we can observe that our algorithm suffers heavily on Cora and Citeseer datasets when the value of θ is set to 0.25. This is because Cora and Citeseer networks are sparse in nature. For $\theta = 0.25$, most of the nodes in the content layer have at most 1 or 2 neighbors. Thus the random walk, though spends more time on the content layer, could not gather complete information about the semantic neighborhood of the nodes. We can see that on Pubmed, better performance is achieved when $\theta \leq 1$, where the trend is mostly opposite of that on Cora and Citeseer. As the link structure of Flickr is not often consistent with its community structure, the performance degrades with increasing values of θ (which forces the random walk to spend more time on the structure layer) on this dataset. This observation is consistent with the results of MIRand and other baseline algorithms for node classification and clustering.

6 DISCUSSION AND FUTURE WORK

In this work, we proposed a novel unsupervised algorithm *MIRand* to embed a graph with content associated with each node. MIRand operates by creating a multilayered network and then employs a random walk driven by the informativeness of a node for learning the network representation. Through experimentation we show the robustness of MIRand in the sense that it is able to intelligently select structure or content layers, specially in the case when one of them is noisy or inconsistent.

In future, we want to conduct experiments on datasets having different types of contents. So far we have only considered static graphs. In contrast, dynamic networks are those that change over time. So we propose to extend our idea to the area of dynamic attributed networks in the future. We also seek to experiment on datasets seeded with outlier nodes to see their effect on the performance of our algorithm for the attributed graph embedding.

REFERENCES

[1] Firoj Alam, Shafiq Joty, and Muhammad Imran, 'Graph based semi-supervised learning with convolution neural networks to classify crisis related tweets', in *Twelfth International AAAI Conference on Web and Social Media*, (2018).

[2] David Arthur and Sergei Vassilvitskii, 'k-means++: The advantages of careful seeding', in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035. Society for Industrial and Applied Mathematics, (2007).

[3] Sambaran Bandyopadhyay, Harsh Kara, Aswin Kannan, and M Narasimha Murty, 'Fscnmf: Fusing structure and content via non-negative matrix factorization for embedding information networks', *arXiv preprint arXiv:1804.05313*, (2018).

[4] Sambaran Bandyopadhyay, N Lokesh, and M Narasimha Murty, 'Outlier aware network embedding for attributed networks', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 12–19, (2019).

[5] Shaosheng Cao, Wei Lu, and Qiongkai Xu, 'Deep neural networks for learning graph representations', in *Thirtieth AAAI Conference on Artificial Intelligence*, (2016).

[6] Hao Chen, Susan McKeever, and Sarah Jane Delany, 'The use of deep learning distributed representations in the identification of abusive text', in *Proceedings of the International AAAI Conference on Web and Social Media*, volume 13, pp. 125–133, (2019).

[7] Brian Gallagher and Tina Eliassi-Rad, 'Leveraging label-independent features for classification in sparsely labeled networks: An empirical study', in *Advances in Social Network Mining and Analysis*, 1–19, Springer, (2010).

[8] Hongchang Gao and Heng Huang, 'Deep attributed network embedding.', in *IJCAI*, pp. 3364–3370, (2018).

[9] Aditya Grover and Jure Leskovec, 'node2vec: Scalable feature learning for networks', in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864. ACM, (2016).

[10] Will Hamilton, Zhitao Ying, and Jure Leskovec, 'Inductive representation learning on large graphs', in *Advances in Neural Information Processing Systems*, pp. 1025–1035, (2017).

[11] William L Hamilton, Rex Ying, and Jure Leskovec, 'Representation learning on graphs: Methods and applications', *arXiv preprint arXiv:1709.05584*, (2017).

[12] Jiaming Huang, Zhao Li, Vincent W Zheng, Wen Wen, Yifan Yang, and Yuanmi Chen, 'Unsupervised multi-view nonlinear graph embedding.', in *UAI*, pp. 319–328, (2018).

[13] Xiao Huang, Jundong Li, and Xia Hu, 'Accelerated attributed network embedding', in *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 633–641. SIAM, (2017).

[14] Thomas N Kipf and Max Welling, 'Semi-supervised classification with graph convolutional networks', in *International Conference on Learning Representations*, (2017).

[15] Preethi Lahoti, Kiran Garimella, and Aristides Gionis, 'Joint non-negative matrix factorization for learning ideological leaning on twitter', in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 351–359, (2018).

[16] David Liben-Nowell and Jon Kleinberg, 'The link-prediction problem for social networks', *Journal of the American society for information science and technology*, **58**(7), 1019–1031, (2007).

[17] Miller McPherson, Lynn Smith-Lovin, and James M Cook, 'Birds of a feather: Homophily in social networks', *Annual review of sociology*, **27**(1), 415–444, (2001).

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, 'Distributed representations of words and phrases and their compositionality', in *Advances in neural information processing systems*, pp. 3111–3119, (2013).

[19] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov, 'Learning convolutional neural networks for graphs', in *International conference on machine learning*, pp. 2014–2023, (2016).

[20] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena, 'Deepwalk: Online learning of social representations', in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710. ACM, (2014).

[21] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang, 'Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec', in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 459–467. ACM, (2018).

[22] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo, '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. ACM, (2017).

[23] Han-Wei Shen, Ling-Jen Chiang, and Kwan-Liu Ma, 'A fast volume rendering algorithm for time-varying fields using a time-space partitioning (tsp) tree', in *Proceedings of the conference on Visualization'99: celebrating ten years*, pp. 371–377. IEEE Computer Society Press, (1999).

[24] Yu Shi, Huan Gui, Qi Zhu, Lance Kaplan, and Jiawei Han, 'Aspem: Embedding learning by aspects in heterogeneous information networks', in *Proceedings of the 2018 SIAM International Conference on Data Mining*, pp. 144–152. SIAM, (2018).

[25] Chenhao Tan, Jie Tang, Jimeng Sun, Quan Lin, and Fengjiao Wang, 'Social action tracking via noise tolerant time-varying factor graphs', in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1049–1058. ACM, (2010).

[26] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei, 'Line: Large-scale information network embedding', in *Proceedings of the 24th International Conference on World Wide Web*, pp. 1067–1077. International World Wide Web Conferences Steering Committee, (2015).

[27] Laurens van der Maaten and Geoffrey Hinton, 'Visualizing data using t-SNE', *Journal of Machine Learning Research*, **9**, 2579–2605, (2008).

[28] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio, 'Graph attention networks', in *International Conference on Learning Representations*, (2018).

[29] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm, 'Deep graph infomax', in *International Conference on Learning Representations*, (2019).

[30] Daixin Wang, Peng Cui, and Wenwu Zhu, 'Structural deep network embedding', in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1225–1234. ACM, (2016).

[31] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo, 'Graphgan: Graph representation learning with generative adversarial nets', in *Thirty-second AAAI conference on artificial intelligence*, (2018).

[32] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu, 'A comprehensive survey on graph neural networks', *arXiv preprint arXiv:1901.00596*, (2019).

[33] Junyuan Xie, Ross Girshick, and Ali Farhadi, 'Unsupervised deep embedding for clustering analysis', in *International conference on machine learning*, pp. 478–487, (2016).

[34] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang, 'Network representation learning with rich text information.', in *IJCAI*, pp. 2111–2117, (2015).

[35] Jiaxuan You, Rex Ying, and Jure Leskovec, 'Position-aware graph neural networks', in *International Conference on Machine Learning*, pp. 7134–7143, (2019).

[36] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung, 'Gaan: Gated attention networks for learning on large and spatiotemporal graphs', *arXiv preprint arXiv:1803.07294*, (2018).

[37] Xinyuan Zhang, Yitong Li, Dinghan Shen, and Lawrence Carin, 'Diffusion maps for textual network embedding', in *Advances in Neural Information Processing Systems*, pp. 7587–7597, (2018).