

Course Project Report

**A Multilayered Informative Random Walk for Attributed Social  
Network Embedding**

*Submitted By*

**Man Mohan Nayak (222IT019)**

**Nikhil Verma (222IT026)**

*as part of the requirements of the course*

**Web and Social Computing (IT752) [Jan-Apr 2023]**

*in partial fulfillment of the requirements for the award of the degree of*

**Master of Technology in Information Technology**

*under the guidance of*

**Dr. Sowmya Kamath S, Dept of IT, NITK Surathkal**

*undergone at*



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA, SURATHKAL**

**APR 2023**


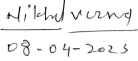
# DEPARTMENT OF INFORMATION TECHNOLOGY

National Institute of Technology Karnataka, Surathkal

## C E R T I F I C A T E

This is to certify that the Course project Work Report entitled **“A Multilayered Informative Random Walk for Attributed Social Network Embedding”** is submitted by the group mentioned below -

### Details of Project Group

Name of the Student	Register No.	Signature with Date
Man Mohan Nayak	222IT019	
Nikhil Verma	222IT026	

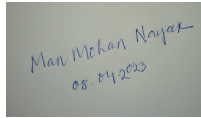
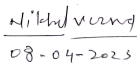
this report is a record of the work carried out by them as part of the course **Web and Social Computing (IT752)** during the semester **Jan-Apr 2023**. It is accepted as the Course Project Report submission in the partial fulfillment of the requirements for the award of the degree of **Master of Technology in Information Technology**.

(Name and Signature of Course Instructor)  
**Dr. Sowmya Kamath S.**

## DECLARATION

We hereby declare that the project report entitled “**A Multilayered Informative Random Walk for Attributed Social Network Embedding**” submitted by us for the course **Web and Social Computing (IT752)** during the semester **Jan-Apr 2023**, as part of the partial course requirements for the award of the degree of Master of Technology in Information Technology at NITK Surathkal is our original work. We declare that the project has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles elsewhere.

### Details of Project Group

Name of the Student	Register No.	Signature with Date
1. Man Mohan Nayak	222IT019	
2. Nikhil Verma	222IT026	

Place: NITK, Surathkal

Date: **08.04.2023**

# A Multilayered Informative Random Walk for Attributed Social Network Embedding

**Abstract**—Network representation learning, sometimes referred to as Graph embedding, a method for mapping a node of a network to a lower level of representation. vector space with dimensions. The efficiency of random walk-based representation techniques is demonstrated by their ease in preserving various orders of vicinity among the nodes of the network in the embedding space. Nowadays, the majority of social networks contain some type of content (or attribute) attached to each node. Along with the network’s link structure, these properties can offer supplementary data. However, there are significant differences between the information carried by the connection structure and that by the properties over the nodes in a real-world network. Most of the unsupervised network embedding algorithms based on attribute is in use today do not differentiate between the informativeness of a node’s attributes and connection structure.

Here, we offer node embedding method which is unsupervised in nature that takes advantage of both structure and characteristics by carefully choosing which one to prioritize in the random walk for every node individually. In order to provide a novel random walk which is combination of the informativeness of a node in various layers, we transform the network to a graph containing multiple layer . This integrated technique is straightforward and computationally quick, and it may employ content to complement structure and vice versa. Experimental analyses on real-world, publicly accessible cora dataset demonstrate the superiority of our method (up to 168.75 improvement) over the leading algorithms in the field.

**Keywords:** embedding, random walk, attribute network

## I. INTRODUCTION

We use social media in every area of our daily lives. Networks are typically represented using graphs. The success of different mining related tasks depends on how well features are chosen and networks are designed. In case of the networks which are sparse and large , as they are in used real world applications, successful feature engineering necessitates a significant amount of topic knowledge and labour. In contrast, in a graph embedding architecture, a function in which each node can be represented in the form of a dense and compact vector is predominantly learned in a task independent manner. algorithms which uses Machine learning techniques work better on these embeddings for network related mining tasks like node categorization, community finding, etc., as shown in the literature. recent outcomes for more social network activities, like classifying tweets relevant to crises and identifying abusive texts show promise for embedding-based techniques.

The preservation of unique order of node vicinity into the embedding space is a key objective of graph embedding algorithms; in other words, nodes that are close to one another in the graph’s local neighborhood should also be

close to each other in the vector space. A random walk applied on a graph creates a succession of nodes where each node within a narrow window is a member of a nearby neighborhood. Naturally, random walk has served as the cornerstone of many well-known graph embedding methods, as demonstrated in . However, the majority of these random walk-based methods merely take the network’s connection topology into account. However, each node in real world social network is coupled with a wealth of qualities (or content). The preservation of distinct orders of node vicinity into the embedding space is a key objective of graph embedding algorithms; in other words, nodes that are close to one another in the graph’s local neighborhood should also be close to one another in the vector space. A random walk on a graph creates a succession of nodes where each node within a narrow window is a member of a nearby neighborhood. Naturally, random walk has served as the cornerstone of many well-known graph embedding methods, as demonstrated in . However, the majority of these random walk-based methods merely take the network’s connection topology into account. However, each node in real world social network is coupled with a wealth of qualities (or content).

While dealing with different network mining tasks node attributes plays a vital role , including resolving the filter bubble problem and creating prediction on social action . Combining attributes of node and structure of link has improved embeddings on nodes in the networks with good quality . However, the majority of the currently used attribute network embedding approaches are either based on matrix factorization or graph neural networks . Due to their linear nature, matrix factorization-based approaches frequently fall short of addressing a network’s extremely non-linear properties. Although graph neural network techniques have been shown to be effective in the literature, they are often semi-supervised in nature since a significant number of the graph neural network parameters must be learned from the labels of the nodes. This is a significant restriction for real world social network since node label, such as a node’s membership in a community, are typically expensive to acquire. Additionally, a recent study on GNNs demonstrates that the two crucial characteristics for any node embedding approach are as follows:

- position aware node embedding is a function that can convert the distances between two nodes’ embeddings into the network’s shortest path distances.

- If a node embedding (for some  $q$ ) is a function of a node's network neighbourhood up to  $q$  hops, it is said to be structure aware.

Since they aggregate neighbourhood attributes, graph neural networks have been shown to be structure aware, but they might not be location aware. Random walk methods can satisfy both of the following conditions: (i) By using negative sampling, the graph distances between two randomly sampled nodes that are close to one another are preserved. This preserves the embeddings of any two randomly chosen nodes. (ii) The node embeddings structure becomes aware after a random walk starting from a node that encompasses several nodes from a  $q$ -hop neighbourhood. (it can cover all nodes if there are additional random walks). In order to accomplish this goal and close the aforementioned research gaps, we provide an unsupervised, random walk based technique that uses the complementarity of the link structure and node characteristics to build node embeddings of a graph.

In real-world networks, it is difficult to incorporate each node's properties into the most recent unsupervised graph embedding algorithms. A node's properties and structure frequently conflict with one another. The semantic information they carry may have considerable gaps. Figure 1 illustrates a situation in which a node's content, which is highlighted in blue, provides more information than its structural elements. For network embedding, it's crucial to take use of a node's level of informativeness in its structure and attribute layers. Though link structure and node characteristics are processed uniformly by current unsupervised attributed network embedding algorithms, this is not the case. When node labels are too expensive to obtain, semi-supervised graph embedding methods like that use attention mechanisms to characterize a node's relevance are challenging to use. In this paper, we address this gap in research.

- We suggest a brand-new unsupervised technique called MIRand, which builds graph with multiple-layer and by using the concept of a brand-new random walk that helps in making the most of a node's informativeness by combining its attributes and structure. According to best of our knowledge, MIRand is the first graph embedding method based on attribute that uses multiple-layered graph random-walk, where one layer represents the structure and the other with the nodes' attributes. As a result, our strategy can be thought of as a fresh extension of node2vec and other random walk based network embedding methods for content networks.

## II. RELATED WORK

In order to get a thorough overview of network embedding, we consult the literature review in [1]. We do, however, discuss some of the most well-known pieces in this section to be thorough. For a very long period, common dimensionality reduction methods or manually created feature engineering methods were used for graph mining jobs. Learning about network representations was impacted by advancements in the field of NLP. By using random walks to create a corpus of node sequences, DeepWalk and node2vec

maximize the log likelihood of each node's context to produce the node embeddings. The 1st order and 2nd order vicinity in node embeddings are explicitly captured in line using two alternative optimization formulas. In [2], methods for network embedding based on autoencoders are proposed. Another random walk-based node embedding method that discovers comparable embeddings for the nodes is Struc2vec (Bandyopadhyay et al., 2018). are similar in structure. In [3], a framework for generative adversarial learning for graph representation was also investigated. Theoretical In [4], analysis is done to demonstrate how matrix factorization and random walk-based embedding methodologies correspond. All of these techniques merely take the network's link structure into account.

The matrix factorization strategy to combine content and link layer for network embedding is initially proposed by TADW (Niepert et al., 2016). The literature also includes joint matrix factorization methods to provide node embeddings for attributed networks based on unsupervised learning [5]. [6] proposes a deep architecture with 2 parallel autoencoders for attributes and structure. The utilization of the relationship between content and structure for embedding of nodes is another deep autoencoder based strategy that is suggested in [7]. [8] develops a matrix factorization-based method for outlier-aware network embedding. By using a network diffusion process, [9] suggests an embedding method meant for content in the form of textual data in a network in addition to the embedding of the nodes. [10]. These methods are unsupervised, but they do not prioritize structure and content differently depending on how they fit into the network.

For attributed graph embedding, semi-supervised (GNN) based approaches are also put forth in latest literature. In [11], a semi-supervised (GCN) model is developed to gather data from neighboring nodes (Yang et al., 2015). [12] suggests GraphSAGE, an extension of GCN that combines neighborhood sub-sampling with several methods of information aggregation. In graph embedding based on attention is established, and expands it to include several attention heads of varying importance. These algorithms can't be used in situations when there is no labeling information available because they are semi-supervised. We present a unique embedding method whose concept is based on unsupervised random walk that, in contrast to them, makes use of a node's informativeness in terms of its content and structure.

## III. PROBLEM STATEMENT

A graph  $G = (E, V, W, F)$  is commonly depicted as an information network, where  $V = (v_1, v_2, \dots, v_n)$  is the collection of nodes (also known as vertexes), each of which represents a data object. The collection of edges connecting the vertexes is denoted by  $E \subseteq (v_i, v_j) | v_i, v_j \in V$ . Each link  $e \in E$  is a pair that is ordered with the coordinates  $e = (v_i, v_j)$  and has a weight  $w_{v_i, v_j} > 0$  that represents the relation's strength. The collection of those weights is called  $W$ .  $G$  will not be directed if  $G$  is unweighted, then  $(v_i, v_j) = (v_j, v_i)$  and

$w_{v_i, v_j} = w_{v_j, v_i}$ ,  $v_i, v_j = 1, w_{v_i, v_j} = 1$ , and  $E.F = f_i | i \in 1, 2, \dots, n$ , where  $f_i \in R_d$  is the content vector connected to node  $v_i \in V$ .  $F$  can therefore be thought of as the content matrix. If the material is text-based,  $F$  can be depicted by a bag-of-words model, where we have to remove stop words and stemming is done as part of the preprocessing. This matrix's rows each represent a tf-idf vector for the information related to text present at the relevant node.  $F$ 's dimension is given by  $d$  and  $n$ , where  $d$  = total number of distinct words (attributes) in the corpus after preprocessing. This generalisation applies to various sorts of content, like images, videos, and audio.

Our goal is to identify a low dimensional vector representation of  $G$ 's nodes that is both compatible with the network's structure and its node content. More specifically, the embedding of network aims to teach a function  $f: v_i \rightarrow x_i \in R^K$  for the given network  $G$ , which transfers each vertex to a vector with  $k$  dimensions where  $K < \min(n, d)$ . This representations should maintain the network's fundamental semantics. Therefore, nodes with similar representations should be those that are near to one another in terms of their distance of position or content similarity. The conventional machine learning methods will perform better on downstream network mining tasks if this representation is also continuous and compact.

#### IV. DATASET

**CORA DATASET:** The 2708 scientific publications that make up the Cora dataset have been divided into seven categories. There are 5429 links in the network of citations. A 0/1-valued vector of word denoting the absence/presence of the matching word from the dictionary is used to describe each publication in the dataset. 1433 distinct words make up the dictionary.

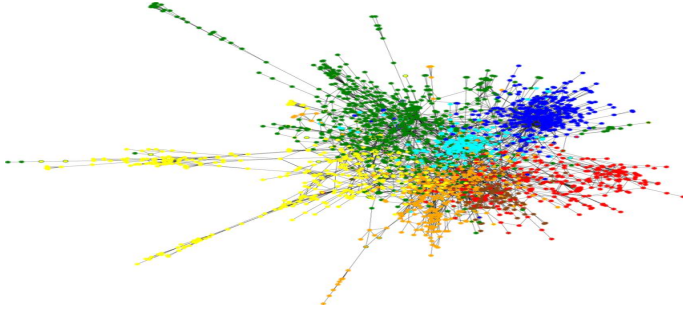


Fig. 1: cora dataset

#### V. METHODOLOGY

##### A. OUR APPROACH

The specifics of our suggested strategy, MIRand, are described in this section. Our first objective is to split the network into two layers the network  $G$  and attribute in the form of a matrix  $F$ . The structure is represented by the first layer, and the content is represented by the second layer.

**Structure Layer:** This layer is conceptually equivalent to the specified network without any attribute in the nodes. The structure layer is  $G_s = (E_s, V_s, W_s)$ , where  $E_s = E$  and

$W_s = W$ , mathematically speaking, given the input network  $G = (E, v, W, F)$ . The only difference between the node set and the supplied network is that the nodes are designated with a superscript "s".

**Content Layer:** This layer is responsible for capturing directed graph how closely two nodes resemble one another in terms of their respective qualities or contents. We define the content layer as the graph  $G_c = G$  as described above.  $(V_c, E_c, W_c)$ . In the content layer, we identify the nodes by adding the superscript "c".  $v \in V \iff v_c \in V_c$ , and  $\forall v \in V$ . On the basis of cosine similarity of the row vectors  $F_i$  and  $F_j$ , it is possible to calculate the similarity or all other nodes' weight  $v_j (j \neq i)$  for each node  $v_i \in V$ .  $w_{ij}^c = \text{Cosine}(F_i, F_j)$ . However, in this instance, the attribute graph would be almost full (that is there is an edge between nearly every pair of vertices), which would lengthen the computation time required to analyse the graph. In contrast, the number of edges in the structural layer is fixed. Therefore, we start by calculating the average number of outgoing edges across all of the structural layer's nodes. Call it  $\text{Avg}_s$  for now.  $\text{Avg}_s = |E|/n$  if the supplied network is directed, and  $2|E|/n$  otherwise if it is undirected. In order to aid the random walk we want the number of edges in the attribute layer to be compared to the number of edges in the structure layer. Therefore, for each node  $v_i^c$  in the content layer, we identify the top  $\theta \times [\text{avg}_s]$  nodes (aside from  $v_i^c$ ) in terms of their content cosine similarity to  $v_i^c$ , where  $\theta$  is referred to as the ratio parameter, or  $\theta \in R_+$ . Then we add a directed edge from  $v_i^c$  with an edge weight equal to the cosine similarity between each node's contents. We also toss out edges with negative weights, which means that the cosine similarity between the 2 end vertices is -ve. According to random walk, determines how long the random walk can stay in a layer. The studies have also demonstrated this. Figure 1 depicts the content and structure layers as well as the connections between them, which will be covered later.

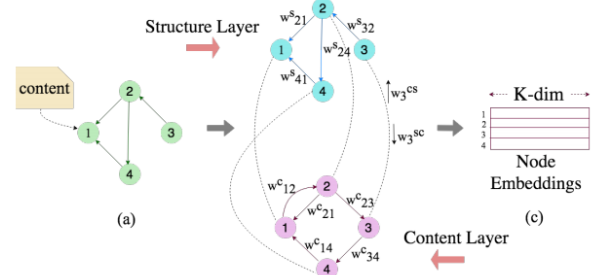


Fig. 2: network with structure and content layer

##### B. Random Walk using Informativeness of nodes in a network

This part, we suggest a unique random walk that would intelligently navigate the nodes in the content and structure layers. As previously said, we start by creating the content

and the structure layers. Next, nodes are connected in a way that , for every node  $v_i$  in graph  $G$  . Consider a network that has the label  $v_i^c$  in the content layer and  $v_i^s$  in the structure network. As will be discussed later, we add 2 weighted and directed edges between  $v_i^s$  and  $v_i^c$ . Our objective is to choose the content or structure layer depends on the node's informativeness  $v_i$  in the corresponding layer whenever the random walk reaches node  $v_i$ . Below, we define the concept of a node's informativeness. Focus on the nodes that are connected with few neighbours so that the next step of a random walk from the current node is having more information. This will help any random-walk for capturing the underlying homophily quality of an content network. In contrast, a node in an information network that has strong connections to many other nodes is actually less resembles to majority of its neighbours. In this work, we take it for granted that an edge connection is strong if its edge weight  $>$  the average weighted edge of the graph layer. Therefore, we define  $S_i$  as the set of neighbours in the structural layer that are closely related to  $V_i^s$ .

$$\Gamma_i^s = \left\{ v_j^s \in V_s \mid w_{(v_i^s, v_j^s)}^s \geq \frac{1}{|E_s|} \sum_{e' \in E_s} w_{e'}^s \right\} \quad (1)$$

Since the average weighted edge of the structure layer1 is greater than the weight of each incoming edge from  $v_i^s$ , each node in  $S_i$  has this edge. The quantity  $|S_i|$  represents how many of  $v_i^s$  structure layer neighbours are strongly related. The content layer's definition for  $\Gamma_i^c$  is similar:

$$\Gamma_i^c = \left\{ v_j^c \in V_c \mid w_{(v_i^c, v_j^c)}^c \geq \frac{1}{|E_c|} \sum_{e' \in E_c} w_{e'}^c \right\} \quad (2)$$

For a random walk, we describe the node's informativeness  $v_i^c$  and  $v_i^s$  respectively, as follows:

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

Let's now take into account the directed edge  $(v_i^s, v_i^c)$  with weight  $w_{ij}^{sc}$ . As previously said, provided that the current node is  $v_i$ , the random walk chooses either the content or the structure layer depending on the informativeness of  $v_i^s$  and  $v_i^c$ . Structure layer selection probability is proportional to  $w_{ij}^{cs}$ , and content layer selection probability is proportional to  $w_{ij}^{sc}$ . We therefore fix these inter layer weights to be  $w_{ij}^{sc} = I_i^c$  and  $w_{ij}^{cs} = I_i^s$ , with  $v_i = V$ . Figure 1 depicts the entire multi-layered network in its entirety. The biased random walk transition probabilities on this multiple -layered graph's inter- and intra-layer boundaries are then defined.

The random walk goes like this. Given that we are at node  $v_i$ , either in the content layer or the structure layer, at a specific time step of the random walk. We first determine the likelihood of moving into the the content layer or structural layer before moving on to the next phase. Moving

to a layer where node  $v_i$  is more informative is our aim. The odds of inter-layer transition are thus defined as:

$$p(v_i^s \mid v_i) = \frac{w_{ij}^{cs}}{w_{ij}^{sc} + w_{ij}^{cs}} = \frac{I_i^s}{I_i^s + I_i^c} \quad (4)$$

$$p(v_i^c \mid v_i) = 1 - p(v_i^s \mid v_i) = \frac{w_{ij}^{sc}}{w_{ij}^{sc} + w_{ij}^{cs}} = \frac{I_i^c}{I_i^s + I_i^c} \quad (5)$$

With smaller value of  $I_i^s$ , greater the number of edges with high edge weights from node  $v_i^s$  in the context of the random walk. Therefore, if the random walk stays in the structure layer, it has numerous options for where to go next. In contrast, if  $I_i^c$  is high, there are fewer out-going edges with significant edge weights coming from the node  $v_i^c$ . The decision is in this instance less random and more informative in the attribute layer than it is in the structure layer for the random walk at node  $v_i$ . Therefore, when  $I_i^c$  is high, our aim is to prioritise the attribute layer, and when  $I_i^s$  is high, our aim is to prioritise the structure layer.

We will now talk about the likelihood of choosing the following vertices from the recent vertex  $v_i$ , assuming that we have already chosen a specific layer as we have just stated. The probabilities of intra-layer transition are thus defined as follows. If the 2nd 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  $l_2$  designate the respective layers  $l_1, l_2 \in s, c$  then it was at node  $v_i^{l_1}$  at time  $t_1$ . Here, we expand the (non normalized) transition probabilities of choosing a node  $v_k^{l_2}$  in layer  $l_2$  (as it is at time  $t$ ) from as follows:

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

The first example takes into account the situation where there is a layer transition ( $l_2 \neq l_1$ ) at time  $t$  and the node  $v_i$  is not directly connected to the layer  $l_2$ 's node  $v_j$ . In this instance, the random walk simply chooses the following node on the basis of weights of the edges that leave the node  $v_j$  in layer  $l_2$  before it. It should be noted that although the reverse may not always be true, if  $l_1 = l_2$ , then  $e_{ij}^{l_2} \in E_{l_2}$ . The unweighted distance between nodes  $v_i^{l_2}$  and  $v_k^{l_2}$  is denoted by the symbol  $d_{ik}^{l_2}$ . Two hyper parameters,  $p$  and  $q$ , are greater than 0. The random walk would be forced to skip the node it previously visited if  $p$  were set to a higher value. Similar to this, a greater  $q$  would prevent the random walk from leaving the neighbourhood of the recently visited node after just one hop. As a result, they can be fixed as indicated in and direct the search to follow a DFS or BFS approach accordingly. The layer information from either layer  $s$  or layer  $c$  is discarded before adding the next node chosen by the procedure to the random walk's sequence.

This guarantees that we only receive one embedding for each node, as opposed to where embeddings for structure and attributes have to be concatenated as a post processing. The number 1 is the random walk's truncated length from a node, and we repeat the previous step 1 times.

We maximise the log-probability of the context of a node in a random walk sequence with regard to that node once the appropriate number of sequences of truncated random walk have been generated for every node. Formally speaking, we wish to increase the following:

$$\max_X \sum_{v \in V} \log P(C(v) | \mathbf{x}_v) = \max_X \sum_{v \in V} \sum_{u \in C(v)} \log P(u | \mathbf{x}_v) \quad (7)$$

$X$  is the collection of all the nodes' embeddings in this case, and  $x_v \in \mathbb{R}^K$  is the node  $v$ 's embeddings. In a random walk, the set of nodes around node  $v$  is known as  $C(v)$ . We also suppose that each node in a context is conditionally independent. The dot product of  $x_u$  and  $x_v$  can be used to parameterize the conventional softmax function, which can be used to represent each of the aforementioned probabilities. The final result of the equation above is the following:

$$\max_X \sum_{v \in V} \sum_{u \in C(v)} (x_u \cdot \mathbf{x}_v - \log Z_v) \quad (8)$$

$Z_v = \sum_{u' \in V} \exp(\mathbf{x}_{u'} \cdot \mathbf{x}_v)$  is the partition function, which negative sampling can be used to approximate. The Eq. 7 optimisation uses stochastic gradient ascent over the model's parameters to solve the problem. Algorithm 1 enumerates MIRand's whole workflow. We have developed a unique kind of heterogeneous network called a two-layered network. But because MIRand takes advantage of the 1-to-1 connection between the content and structural representations of the same node and extracts the information which is complementary, our technique naturally differs from any heterogeneous network embedding.

Please be aware that MIRand is easily adaptable to social networks with several content types (for instance, Facebook users who submit texts, photos, and videos). If so, we can have more than two layers (one for content and then one for each sort of structure) and use intra- and inter-layer transition probability to perform a random walk across them all.

## VI. OBSERVATION AND RESULT

**1. t-SNE node visualization** for evaluating our model first we are performing network graph visualization in this we differentiate node in communities in the graph network. for this task the embedding created by MIRand followed by word2vec model, in which each node attribute is represented by a 1 X 128 dimension vector. in the graph visualization we can clearly observe different community which were not clear in visualization created by other embeddings using only link structure of the graph.

**Node classification task** for this task we have used the embedding created by random walk using MIRand

---

### 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  $K \ll \min(n, d)$ ,  $\theta$ : 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  $iter \in \{1, 2, \dots, r\}$  do
4:   for  $v \in V$  do
5:      $Walk = [v]$  ▷ Initialize the Walk (sequence of nodes)
     for the truncated random walk
6:       for  $walkIter \in \{1, 2, \dots, l\}$  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

```

---

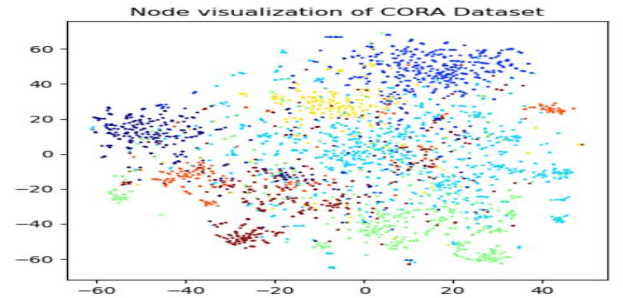


Fig. 3: Node visualization of cora dataset

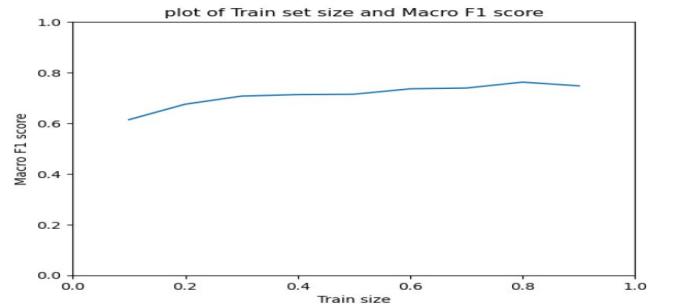


Fig. 4: macro F1 score of cora dataset



as the dataset. the cora dataset contains publication of 6 classes namely – genetic algorithm, neural network, probabilistic methods, case based, reinforcement learning, and rule learning. for classification we have machine learning technique called Random Forest Classifier which creates multiple decision trees of varying sample size of the training data and for prediction vote is taken from decision tree and class with maximum votes is decided as output.

for training and testing the classifier model we have use different split of dataset from 10 % for training and 90% for testing to 90 % for training and 10 % for testing. the highest accuracy we got for training size 80 % and test data size 20 %. we got classification accuracy of 78.04% which is better than most baseline algorithms.

we have also plotted Macro F1 score and Micro F1 score with percent of data used for training. in the plot we observe very good results that are better than SDNE, AANE, GraphSAGE, Struc2Vec, DGI embedding technique using only the link structure layer for graph embedding.

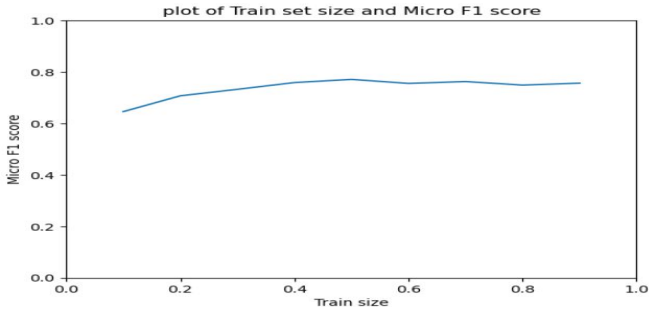


Fig. 5: micro F1 score of cora dataset

future work – in the implementation we have used link structure layer and one attribute layer. as network node can many attributes to take then into account in creating the node embedding we can use multiple attribute layer for generating random walk. using multiple attribute layer we will model real world network more accurately.

## VII. CONCLUSION AND FUTURE WORK

This paper discusses , a brand-new unsupervised technique called MIRand was presented to embed a graph with attribute assigned to every node. In order to learn the network representation, MIRand first builds a multiple-layered network and then uses a random walk that is informed by the informativeness of each node. We demonstrate the robustness of MIRand through testing, namely in the ability to choose structure or content layers sensibly when one of them is noisy or inconsistent. Future tests will be performed on datasets with various content kinds. We have only thought about static graphs thus far. Dynamic networks, on the other hand, are networks that alter over time. Therefore, in the future, we suggest expanding our concept to include dynamic attributed networks. We also want to test our algorithm’s effectiveness

for the attributed graph embedding on datasets that have been seeded with outlier nodes to determine how they affect the results.

## REFERENCES

- Bandyopadhyay, S., Lokesh, N., and Murty, M. N. (2018). Outlier aware network embedding for attributed networks. *CoRR*, abs/1811.07609.
- Niepert, M., Ahmed, M., and Kutzkov, K. (2016). Learning convolutional neural networks for graphs.
- Yang, C., Liu, Z., Zhao, D., Sun, M., and Chang, E. Y. (2015). Network representation learning with rich text information. In *Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI’15*, page 2111–2117. AAAI Press.

# APPENDIX

## WSC

### ORIGINALITY REPORT

14%

SIMILARITY INDEX

12%

INTERNET SOURCES

13%

PUBLICATIONS

0%

STUDENT PAPERS

### PRIMARY SOURCES

1

[eprints.iisc.ac.in](https://eprints.iisc.ac.in)  
Internet Source

11%

2

"ECAI 2020", IOS Press, 2020  
Publication

2%

3

"Database Systems for Advanced  
Applications", Springer Science and Business  
Media LLC, 2019  
Publication

<1%

4

Submitted to Imperial College of Science,  
Technology and Medicine  
Student Paper

<1%

5

[mdpi-res.com](https://mdpi-res.com)  
Internet Source

<1%

6

[dokumen.pub](https://dokumen.pub)  
Internet Source

<1%

7

Shweta Garg, Ramasuri Narayanam,  
Sambaran Bandyopadhyay. "A framework to  
preserve distance-based graph properties in  
network embedding", Social Network Analysis  
and Mining, 2022

<1%