# Attributed Network Representation Learning

***A B. Tech Project Report Submitted***
***in Partial Fulfillment of the Requirements***
***for the Degree of***

**Bachelor of Technology**

*by*

**Anket Kotkar**
(180101037)

*under the guidance of*

**Dr. Sanasam Ranbir Singh**


**to the**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI**
**GUWAHATI - 781039, ASSAM**

# Acknowledgements

I would like to express my gratitude towards **Dr. Sanasam Ranbir Singh** for his timely guidance and encouragement to pursue the problem statement for my project. I would also like to thank Anasua Mitra for her timely support towards my project.

# Contents

**List of Figures** **5**

**List of Tables** **6**

**1 Introduction** **1**

1.1 Necessity of Analysis . . . . . 1

1.2 Network Representation Learning . . . . . 2

1.3 Organization of Report . . . . . 3

**2 Previous Works** **4**

2.1 DeepWalk . . . . . 4

2.2 Node2Vec . . . . . 5

2.3 Graph Convolutional Network . . . . . 6

2.4 GraphSAGE . . . . . 7

2.5 DANE . . . . . 8

2.6 GraphBERT . . . . . 9

2.7 Conclusion . . . . . 10

**3 Problem Formulation and Elementary Experimentation** **11**

3.1 Motivation . . . . . 11

3.2 Problem Formulation . . . . . 11

3.3 Experiments . . . . . 12

3.3.1 Graph without Attributes . . . . . 12
3.3.2 Graph with Attributes . . . . . 13
3.4 Conclusion . . . . . 13

**4 Conclusion and Future Work** **14**

**References** **15**

# List of Figures

2.1 DeepWalk - Hierarchical Softmax . . . . . 5
2.2 Random Walk Procedure in Node2Vec . . . . . 6
2.3 Graph Convolutional Network . . . . . 7
2.4 GraphSAGE . . . . . 7
2.5 DANE Architecture . . . . . 8
2.6 GraphBERT Architecture . . . . . 9

# List of Tables

3.1 Datasets . . . . . 12
3.2 GraphBERT on unattributed network . . . . . 12
3.3 GraphBERT on attributed network . . . . . 13

# Chapter 1

# Introduction

Graphs occur naturally in many different real-life contexts, such as social media networks, communication networks, citation networks or protein-protein interactions. A network can be visualised to capture different kinds of entities and their relations or interactions. These entities and links can be homogeneous as well as heterogeneous, independent of each other. The network provides a more generalised way of representing interactions observed in a real-world scenario.

## 1.1 Necessity of Analysis

Relations between different entities play an essential role in building social structures, information diffusion, news spread and communication patterns, influence on choices of individuals. Modelling these inter-relations as a network helps researchers to do a systematic study to understand the systems. Network analysis is essential for downstream applications across disciplines like targeted advertising or user suggestions in social media networks, possible disease cures in protein interaction networks, or personalised recommendations in different service providers.

Network analysis tasks greatly depend on how the network is represented. Often an adjacency matrix is used to describe the network for analysis. But the order of magnitude creates a challenge. Often these networks are of scale hundreds of millions to even billions of nodes and edges. In these cases, matrix operations which have time complexity of quadratic in the number of vertices, make

them very expensive or sometimes even computationally impossible. Furthermore, a straightforward adjacency matrix representation cannot capture high-level constructs and complex, highly non-linear network structure and vertex or edge attributes. To tackle these difficulties, recently, Network Representation Learning has been seen as a revolutionary way.

## 1.2 Network Representation Learning

Network Representation Learning, also called NRL, tries to learn latent low dimensional vector representations for the network nodes to preserve topological space proximity. Attributed Network Representation Learning tries to learn low dimensional latent representations to preserve topological space proximity and attribute space proximity. Low-level representations of nodes are called embeddings. By preserving the topological space proximity, we mean that 2 nodes with similar topological structures should have similar embeddings. This embeddings similarity can be measured either by euclidean distance or cosine similarity. These learned embeddings are used as an input to the downstream tasks making their computation practically feasible.

The network Representation Learning field has seen significant advancements in network processing algorithms. Initial methods in the NRL field were devised only to consider the network topology and thus embed the nodes with similar neighbourhood structures close into embedding space. But apart from just the topological structure, node and edge attributes play a crucial role in how these networks evolve or are created. Gradually, the importance of using node attributes and labels if available in network learning has been realised. Recently proposed models based on graph neural networks or a transformer-based architecture that takes network features as input has proved to perform better than the existing SoTA architectures on many downstream tasks. We discuss this in detail in chapter 2.

While considering attributed networks, many existing models use the node features at the input layer only. GraphBERT is one example of it. There has seen significant progress on this line for Graph Neural Network based methods. But in GraphBERT, there has not been any such work on this line. As part of my thesis project, I will investigate how to include these attributes at the input

layer along with network topology generated features and use the node attributes at higher layers.

## 1.3 Organization of Report

This chapter provides an overview of the topics covered in the rest of the chapters. We described motivation and necessity for the network analysis. Then we offered an introduction to what Network Representation Learning stands for. The rest chapters are organised as follow: the next chapter provides background regarding existing algorithms, the third chapter discusses problem formulation and basic experimentation, and at last, we discuss the future plans.

# Chapter 2

# Previous Works

Network Representation Learning has seen significant advancements in recent times. In this chapter, we will go through some of the network representation learning techniques.

## 2.1 DeepWalk

In NLP, the Skipgram model has shown great success, which uses sequences of words to learn the word embeddings. Random walk-based methods adopt that analogy in graph context by considering graph as a vast corpus of nodes and generating random walks among them using their interlinks analogous to word sequences. Deepwalk[1], a random walk based method, generates the random walks by considering the next node uniformly randomly from the neighbourhood of the node currently under consideration. After these sequences are sampled, Skipgram is applied to them to learn the node embeddings. But softmax in Skipgram is very expensive. For that, Deepwalk introduces hierarchical softmax as in fig 2.1. In this, when considering maximising probability between node v3 to occur in the context of node v1, we maximise the likelihood of path going from root to v3 when the representation of v1 is given as input. For hierarchical softmax, all the nodes of the graph are considered nodes of a complete binary tree, and every node except at the last layer is a binary classifier, choosing to go to left or right. But DeepWalk has the limitation of only considering the network structure. It does not include the node attributes, thus leaving a lot of useful information.

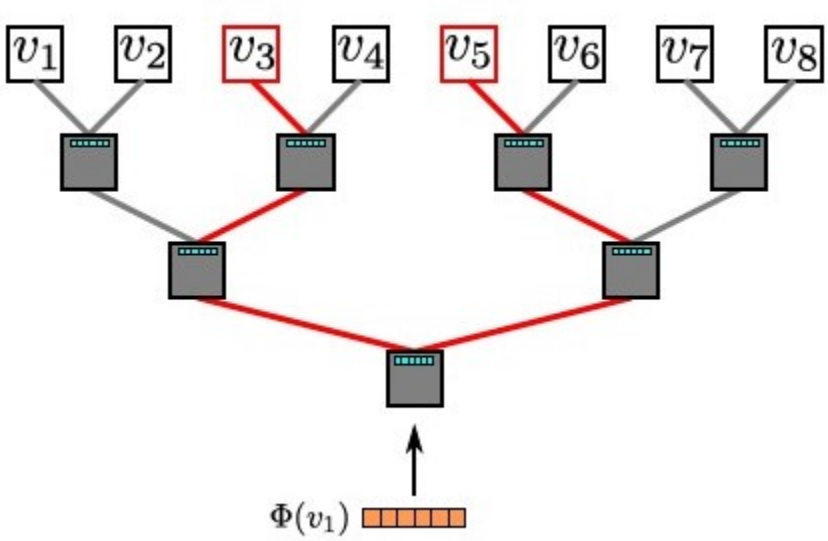


**Fig. 2.1** DeepWalk - Hierarchical Softmax

## 2.2 Node2Vec

Deepwalk is also constrained by a very rigid selection strategy for the next node in the walk. Node2Vec[2] proposes a flexible way of sampling subsequent nodes in the walks using 2 tunable parameters $p$ and $q$. In Node2Vec, biased random node sequences are created using the modified probability of transition between nodes. For a walk with the current node as v with the previous node as t, the probability of going to node x from node v depends on the distance of x from t as shown from fig 2.2 where the shown edge weights act as weights for the original transition probability between v and x.

After generating walks using this transition probability matrix, the Skipgram model is applied to generate the node embeddings. Node2Vec uses negative sampling along with stochastic gradient descent for calculating softmax. This technique of tunable parameters for random walk generation gives a competitive edge over other models. But Node2Vec poses the limitation of considering the topological structure only. It does not include the node attributes, thus leaving a lot of helpful information.

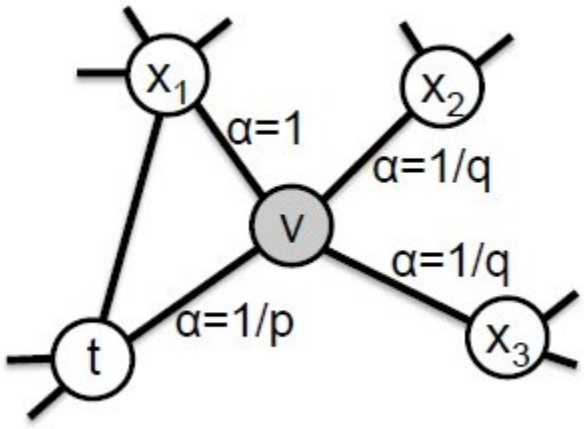


**Fig. 2.2** Random Walk Procedure in Node2Vec

## 2.3 Graph Convolutional Network

Convolutional Networks, a significant advancement in graph neural networks, give a great way in which information from the surrounding nodes can be collected using stacked convolutional filters. Graph Convolutional Network (GCN)[3] takes their motivation from $1^{st}$ order approximations of the spectral convolutions of the graph. GCN takes input adjacency matrix A and network feature matrix X for a semi-supervised learning process. GCN progresses according to the update rule

$$H^{l+1} = \sigma(\tilde{D}^{\frac{-1}{2}}\tilde{A}\tilde{D}^{\frac{-1}{2}}H^{l}W^{l})$$

where $\tilde{A} = A + I$, $\tilde{\mathrm{D}}$ is the degree matrix of the $\tilde{\mathrm{A}}$, and $\sigma$ is the non-linearity function. The loss function for GCN consists of both supervised loss and graph regularisation loss. Thus in backpropagation, the supervised loss from labels is also propagated to unlabeled nodes, which improves the learning process. But GCN has problems of suspended animation and over-smoothing, which prevents deep GCN models. But deep networks are necessary to learn highly non-linear information of network structure and attributes. GCN also causes problems while training due to memory intensive training procedures.

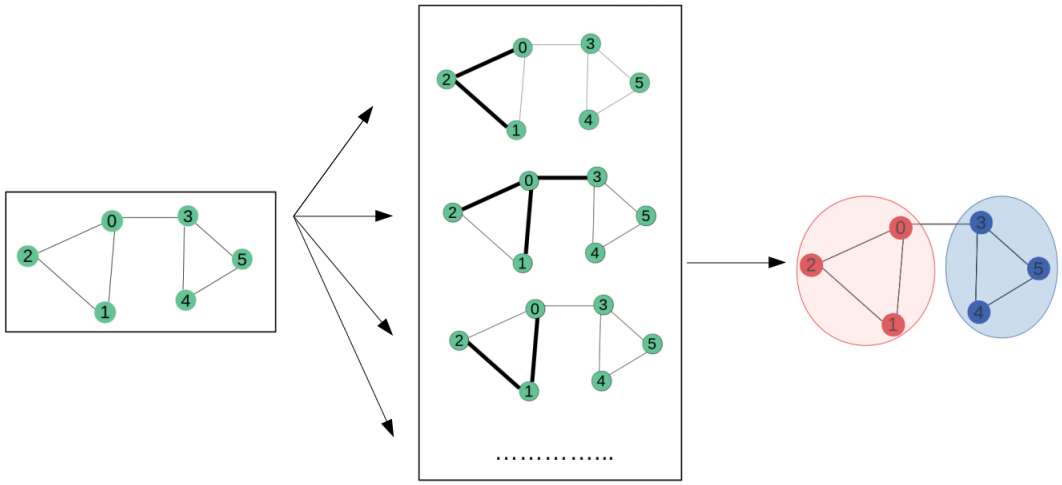

**Fig. 2.3** Graph Convolutional Network

## 2.4 GraphSAGE

GraphSAGE[4] is another graph convolutional network which is based on spatial propagation-based technique. Like GCN, GraphSAGE also take node features as the input layer embeddings. Instead of using rigid matrices as in GCN presented earlier, GraphSAGE uses aggregation functions at each hidden layer to calculate the aggregated embeddings from the Embedding of the current node and the nodes in the sampled surrounding the current node. These aggregated embeddings are then passed on to the successive layers by multiplying with weight matrices. Aggregation functions used

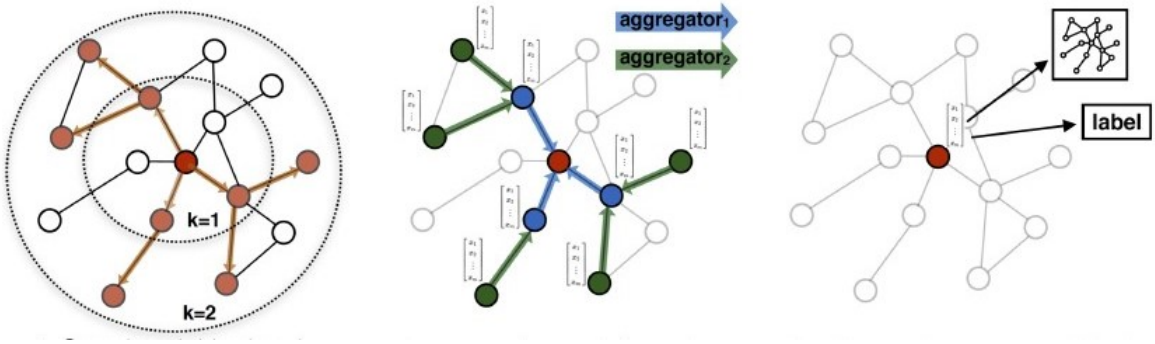


**Fig. 2.4** GraphSAGE

can be mean aggregator, LSTM aggregator or pooling aggregator. GraphSAGE is a good inductive

model, as its trained aggregator functions can be applied to unseen graphs. It has some limitations like uniform sampling strategy, not applicable to directed graphs and high computational power necessity.

## 2.5 DANE

For feature learning, autoencoder acts a powerful unsupervised deep model. The basic autoencoder contains a input layer, then k hidden encoding layer, then k hidden decoding layer and a output layer such that output layer embeddings are of same dimension as input and then reconstruction loss is considered for the refining the parameters. DANE[5] is a deep autoencoder based model. It

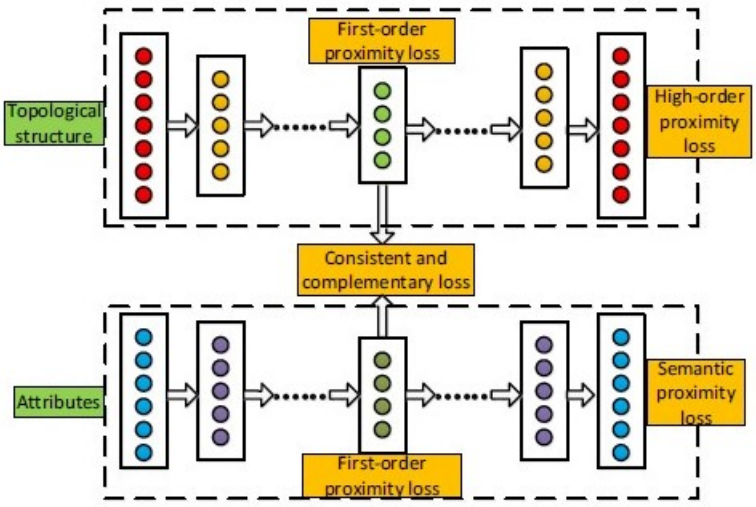


**Fig. 2.5** DANE Architecture

tries to preserve the attribute space proximity, first-order proximity and high order proximity. As attributes and network topology are 2 heterogeneous sources of information about the same nodes, it preserves their distinct identities. Still, it simultaneously keeps in mind that they represent the same identity. The hidden layer embeddings from autoencoding of attributes and network structure are concatenated at last to get the final representation of the nodes.

## 2.6 GraphBERT

GraphBERT[6] takes its motivation from the highly successful BERT model in NLP. All previous NRL models consider the node links. But GraphBERT, apart from the subgraph batching, does link less graph learning. This independence from node links makes GraphBERT adaptable to any graph. GraphBERT takes input 4 kinds of inputs, namely feature embeddings, Weisfeiler-Lehman role embedding, intimacy based positional embedding and hop based distance embedding. GraphBERT first samples subgraphs from the original graph based on the intimacy scores obtained from the Intimacy matrix, calculated using the PageRank algorithm. Using these subgraphs, and the 4 input embeddings for the nodes in this subgraph, GraphBERT, then, by computing through several hidden layers, generates the final embeddings. As GraphBERT needs feature vector as input, it can be applied to non attributed networks by generating embeddings using topological structure using some other method.

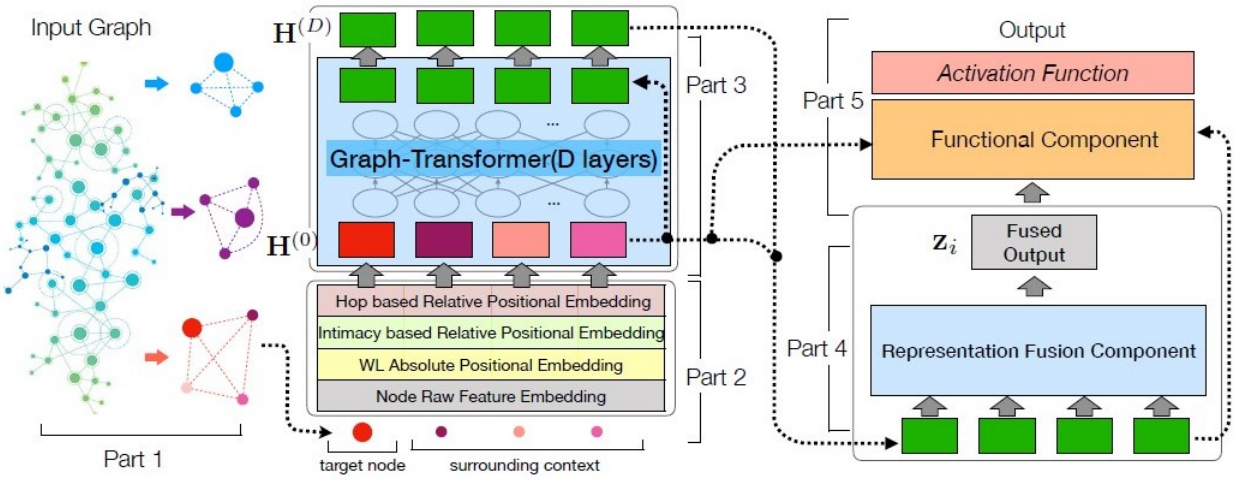


**Fig. 2.6** GraphBERT Architecture

One thing that is missing in GraphBERT is that node features are not used at higher layers. There are no studies regarding this, to the extent of my literature survey and the current date of the project. So I wish to explore this area.

## 2.7 Conclusion

This chapter provided a brief summary of some of the existing algorithms for network representation learning tasks. NRL algorithms have progressed to include node attributes too, as they are an inherent part of network. Most models today are Transformer or GNN based. In the next chapter, we will discuss the formulation of the problem statement by considering the limitations of the current literature.

# Chapter 3

# Problem Formulation and Elementary Experimentation

## 3.1 Motivation

Node attributes are an integral part of the way networks are built. The inclusion of node attributes has proven to improve the performance of the different models. GCN is an excellent example of that. In GCN, node attributes are used at the input layer and through the hidden layers. Using attributes at higher layers maintains the original node identity as well as the derived node identity.

## 3.2 Problem Formulation

In GraphBERT, we need to provide raw node feature vectors. For unattributed networks, we only have the network structure. So we need to generate node embeddings from some other NRL technique by giving input as the network structure which are then used as raw node attributes. But for applying GraphBERT on attributed networks, we have network structure, and node attributes to generate the raw feature vectors. Through this project, I propose investigating how node attributes can be included in the input layer of the GraphBERT along with network structure and the higher layers of the GraphBERT.

**Problem Statement:** To investigate how node attributes can be included at the input layer along with network structure and also in the higher layers while applying GraphBERT on attributed Networks.

As for my project, I have chosen to work on GraphBERT. Here I present some of my experiments with GraphBERT on some attributed networks.

## 3.3 Experiments

In experiments, I have considered datasets Cora, Citeseer. Statistics regarding these datasets is given in the following table. Experiments are conducted for node classification task.

| Dataset | Nodes | Edges | Attribute Dimension |
|---|---|---|---|
| Cora | 2780 | 5429 | 1433 |
| Citeseer | 3312 | 4732 | 3703 |

**Table 3.1** Datasets

### 3.3.1 Graph without Attributes

First, to experiment with, I considered datasets without their node attributes. In this case, I generated node embeddings by running unsupervised Deepwalk on the network, which only considers topological structure. These embeddings were fed into the GraphBERT as node raw features. Results for them are enlisted in the table.

| Dataset | Embedding Dimension | Accuracy |
|---|---|---|
| Cora | 256 | 0.702 |
| Citeseer | 256 | 0.477 |

**Table 3.2** GraphBERT on unattributed network

### 3.3.2 Graph with Attributes

Here I considered node attributes that were generated from meta-information regarding nodes. For the considered datasets, this meta-information is in the form of textual data. The results are listed in the following table.

| Dataset | Accuracy |
|---|---|
| Cora | 0.840 |
| Citeseer | 0.712 |

**Table 3.3** GraphBERT on attributed network

## 3.4 Conclusion

In this chapter, we discussed the motivation behind the problem statement and formally defined the problem that I am trying to investigate through this project. In the next chapter, we will present a conclusion and plans for future work.

# Chapter 4

# Conclusion and Future Work

From the experiments carried out in previous chapter, it is clearly visible that GraphBERT performs better when node attributes are given as raw feature input compared to when features are generated from other NRL technique using network topology itself. As the part of Phase - II of BTP, I would like to explore various strategies in which the node attributes can be included at higher levels in GraphBERT and if network topology generated features can be included at the input layer.

# References


[1] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk," *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, Aug 2014.

[2] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," 2016.

[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017.

[4] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," 2018.

[5] H. Gao and H. Huang, "Deep attributed network embedding," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 3364–3370, International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[6] J. Zhang, H. Zhang, C. Xia, and L. Sun, "Graph-bert: Only attention is needed for learning graph representations," 2020.