UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

An application of graph neural networks on topic modelling with bi-partite graphs

Bruno Fernandes Bessa de Oliveira

Dissertação de Mestrado do Programa de Mestrado Profissional em Matemática, Estatística e Computação Aplicadas à Indústria (MECAI)

ICMC USP SÃO CARLOS

Bruno Fernandes Bessa de Oliveira

An application of graph neural networks on topic modelling with bi-partite graphs

Master dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC- USP, in partial fulfillment of the requirements for the degree of the Master – Professional Masters in Mathematics, Statistics and Computing Applied to Industry. *FINAL VERSION*

Concentration Area: Mathematics, Statistics and Computing
Advisor: Prof. Dr. Alneu de Andrade Lopes

USP – São Carlos
October 2023

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

O48a

Oliveira, Bruno Fernandes Bessa de
An application of graph neural networks on topic modelling with bi-partite graphs / Bruno Fernandes Bessa de Oliveira; orientador Alneu de Andrade Lopes. -- São Carlos, 2023.
92 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Mestrado Profissional em Matemática, Estatística e Computação Aplicadas à Indústria) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2023.

1. Deep learning com grafos. 2. Aprendizado de máquina. 3. Redes neurais. 4. Modelagem de tópicos. I. Lopes, Alneu de Andrade, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Bruno Fernandes Bessa de Oliveira

Uma aplicação de redes neurais de grafos em modelagem de tópicos com grafos bipartidos

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre – Mestrado Profissional em Matemática, Estatística e Computação Aplicadas à Indústria. *VERSÃO REVISADA*

Área de Concentração: Matemática, Estatística e Computação

Orientador: Prof. Dr. Alneu de Andrade Lopes

USP – São Carlos
Outubro de 2023

To Maíra

ACKNOWLEDGEMENTS

To my advisor Dr. Alneu de Andrade Lopes for introducing me to the exciting world of complex networks and graph neural networks and for the necessary guidance to conduct the project. This degree is an old desire of mine, and I very much appreciate you for giving me this opportunity.

To my colleagues in the research group, especially Nicolas Roque, for his support in conducting experiments. To Rafael Rossi for suggestions of benchmark datasets. To Dr. Thiago Faleiros, from the University of Brasilia, for the help in handling the topic propagation model.

To my undergraduate friends, especially Daniel Pizetta, Gilberto Volpe, Danilo Silva, Felipe Camargo and Thiago Bartanha for nourishing our bond and for giving me the motivation to thrive on this path. I wish you know how much I admire you and how important your friendship is after these many years since we shared the São Carlos USP Campus. To Guilherme Melazi for the support and always insightful conversations.

To my analyst Marcelo Tomassini for all the support on dealing with my subjectivity.

To my former managers at Pagbank Marco Pedro, Claudio Moreira and Francisco Moura for being comprehensive during the time I had to take classes for this program. I hope that more people can have such good bosses like you in the industry.

To my family for all the sacrifices made to me getting an education. Each one of you is precious and unique. Our lives have been far from easy coming from where we all started. I dedicate this achievement to your effort and companionship.

To all the professors I had classes from and could be inspired by. To the professionals of the ICMC post-graduate office for institutional service, specially Monique da Conceição.

To the University of São Paulo for being such an awesome institution to which I am honored to come back 10 years after finishing my Bachelor's. I pledge to use this degree to make society better and to defend the legacy of Public Education against the evils of obscurantism, pseudoscience and fascism.

My special thanks to my wife Maíra Cilotti. You know that I could not have made it this far without you. Thank you for lifting me up the many times I needed. I love you.

“Grandes são os desertos, e tudo é deserto.”
(Álvaro de Campos)

RESUMO

OLIVEIRA, B. F. **Uma aplicação de redes neurais de grafos em modelagem de tópicos com grafos bipartidos**. 2023. 92 p. Dissertação (Mestrado – Mestrado Profissional em Matemática, Estatística e Computação Aplicadas à Indústria) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

Grafos são estruturas de dados adequadas para representar objetos do mundo real e suas inter-relações, tendo sido amplamente estudados teoricamente e com múltiplos exemplos de aplicações na indústria e pesquisa acadêmica. A aplicação de dados originados de grafos em aprendizados de máquina teve um significante avanço com a proposta das Redes Neurais de Grafos (*Graph Neural Networks*, ou *GNNs*), permitindo a representação deste tipo de dados em algoritmos que são capazes de preservar as características do grafo sem necessidade de pré processamento. Nesta dissertação apresentamos uma análise das redes neurais de grafos e uma proposta de aplicação no contexto de classificação de textos utilizando modelagem de tópicos para criação de variáveis descritivas em grafos bipartidos.

Palavras-chave: Deep learning com grafos, aprendizado de máquina, redes neurais, modelagem de tópicos.

ABSTRACT

OLIVEIRA, B. F. **An application of graph neural networks on topic modelling with bi-partite graphs**. 2023. 92 p. Dissertação (Mestrado – Mestrado Profissional em Matemática, Estatística e Computação Aplicadas à Indústria) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2023.

Graphs are data structures proper to represent real-world objects and their relationships having been widely studied in theory and with multiple examples of applications in industries and academic research. Applying graph-based data in machine learning had a significant advance with the proposal of Graph Neural Networks (GNNs), allowing the representation of this type of data in algorithms that can retain features from the graph without the need for preprocessing stage. This master's dissertation presents an analysis of GNNs and proposes an application on text classification using topic modelling to create descriptive variables in bi-partite graphs.

Keywords: Graph deep learning, machine learning, neural networks, topic modelling.

LIST OF FIGURES

Figure 1 – Visual representation of a regular square lattice undirected graph 29
Figure 2 – Bipartite graph with randomly connected nodes 31
Figure 3 – Visual representation of graph node features 31
Figure 4 – Rosenblatt's Perceptron 32
Figure 5 – Pair of linear separable patterns (a) and pair of non-linear separable patterns (b) 33
Figure 6 – Multi-layer perceptron with n fully connected hidden layers 34
Figure 7 – Node embedding problem 37
Figure 8 – Neural Message Passing Diagram 40
Figure 9 – Basic pipeline for GNN modelling 41
Figure 10 – Multi-layer Graph Convolutional Network (GCN) with first-order filters . . 45
Figure 11 – Transformer Model 48
Figure 12 – Graph attention layer 49
Figure 13 – Plate notation for LDA 57
Figure 14 – PBG representation of text 59
Figure 15 – Graph Topic Model diagram 61
Figure 16 – Text processing for the proposed model 64
Figure 17 – TPBG training 64
Figure 18 – Bi-partite graph representation of text 65
Figure 19 – Bi-partite GNN learning 66

LIST OF TABLES

Table 1 – List of common activation functions for MLP 35
Table 2 – Summary of shallow embedding methods 38
Table 3 – List of benchmark datasets 67
Table 4 – List of experimental parameters 68
Table 5 – Experimental results for dataset 20 News Groups 69
Table 6 – Experimental results for dataset BBC News 70
Table 7 – Experimental results for dataset Classic4 71
Table 8 – Experimental results for dataset NSF 72
Table 9 – Experimental results for dataset WebKB 73
Table 10 – Experimental results for dataset CSTR 74
Table 11 – Experimental results for dataset Re8 75
Table 12 – Experimental results for dataset DMOZ Computers 76
Table 13 – Experimental results for dataset DMOZ Health 77
Table 14 – Experimental results for dataset DMOZ Science 78
Table 15 – Experimental results for dataset DMOZ Sports 79
Table 16 – Classification F1-score (%) Comparison 80

LIST OF ABBREVIATIONS AND ACRONYMS

BERT	Bidirectional Encoder Representation Transformers
CE	Cross Entropy Loss Function
CSTR	Computer Science Technical Reports Collection Dataset
DEC	Decoder
DMOZ	Directory Mozilla The Open Directory
ENC	Encoder
FL	Focal Loss Function
GAT	Graph Attention Network
GATON	Graph Attention Topic Modeling Networks
GCN	Graph Convolutional Network
GIN	Graph Isomorphism Network
GNN	Graph Neural Network
GNTM	Graph Neural Topic Model
GPT	Generative Pre-trained Transformers
GraphSAGE	Graph Sample and Aggregate
GTM	Graph Topic Model
GTN	Graph Transformer Network
IR	Information Retrieval
LDA	Latent Dirichlet Allocation
LSA	atent Semantic Analysis
LSI	Latent Semantic Indexing
MLP	Multi-layer Perceptron
n-WL	n-th Weisfeiler-Lehman algorithm
NMF	Nonnegative Matrix Factorization
NSF	National Science Foundation Collection Dataset
PBG	Propagation on Bi-partite Graphs
PGM	Probabilistic Graph Model
pLSI	Probabilistic Latent Semantic Indexing
SVM	Singular Value Decomposition
TF-IDF	Term frequency-inverse document frequency score
TM	Topic Modelling

TPBG	Transductive Propagation on Bi-partite Graphs
WebKB	Web Knowledge Base Dataset

LIST OF SYMBOLS

G — Graph

v — Node

e_{ij} — Edge beteween nodes v_i and v_j

$|V|$ — Number of nodes in G

$\mathcal{N}(v_i)$ — Adjacency of node v_i

$a, \mathbf{a}, \mathbf{A}$ — Scalar, Vector and matrix

$\mathscr{L}$ — Laplacian matrix

φ — Non linearity, activation function

η — Neural network learning rate

$h_i^{(k)}$ — GNN representation of node v_i at k-th hop

$\mathscr{V}$ — Corpus vocabulary

$Poisson$ — Poisson distribution

Dir — Dirichlet distribution

F_1^{μ} — F1 micro-average score

$\mathcal{O}$ — Order of complexity of an algorithm

CONTENTS

1 INTRODUCTION . . . 25
1.1 Motivation . . . 25
1.2 Applications . . . 27
1.3 Objectives and Methods . . . 27
1.4 Contributions . . . 28
1.5 Structure of this Work . . . 28

2 BACKGROUND . . . 29
2.1 Graphs . . . 29
2.2 Graph Semantic Information . . . 31
2.3 Neural Networks and Deep Learning . . . 32
2.4 Prior Approaches for Machine Learning with Graphs . . . 36
2.5 Graph Neural Networks . . . 38
2.6 Summary . . . 41

3 VARIANTS AND APPLICATIONS . . . 43
3.1 Spectral Methods . . . 43
3.2 Convolutional GNN Variants . . . 44
3.3 Spatial Methods . . . 47
3.4 Attention-based Methods . . . 47
3.5 Applications . . . 51
3.6 Model Interpretability . . . 52
3.7 Computational Implementations . . . 53
3.8 Summary . . . 54

4 TOPIC MODELLING . . . 55
4.1 Topic Modelling Techniques . . . 55
4.2 Pre-trained Models . . . 58
4.3 Graph Representation for Topic Modelling . . . 58
4.4 GNN for Topic Modelling . . . 60
4.5 Summary . . . 62

5 PROPOSAL AND RESULTS . . . 63
5.1 Proposed Model . . . 63

5.2 Experimental Setup 66
5.3 Experimental Results 69
5.4 Discussion 80
5.5 Complexity Analysis 82
5.6 Summary 82

6 CONCLUSION 83

BIBLIOGRAPHY 85

CHAPTER

1

INTRODUCTION

1.1 Motivation

Data structures are vital for computational problem solving because they provide a proper way to represent information about the real world to be handled efficiently by computational algorithms. The study field of *Complex Networks* explores the properties of graphs applied all across the sciences. To name a few uses of graphs: model interactions of atoms and molecules (Physics, Chemistry, Biology), power grids (Energy Supply), the World Wide Web (Computer Science), the dynamics of diseases (Epidemiology), citation in scientific production, dynamics of social media platforms (Social Sciences), networks of words.

Machine learning is the study of software that is able to perform tasks with performance improved by learning from experience (data sample fed to an algorithm) (MITCHELL, 1997). The past two decades experienced a fast-paced development of *Neural Networks*, a particular class of classic machine learning algorithms following the development of the *Back-propagation* algorithm (RUMETHART, 1986) that allowed the modelling of neural networks that use Rosenblatt's multi-layer perceptron in an arbitrary number of layers for general-purpose task solving, taking advantage of high-performance hardware and GPUs, then leading to the rise of *Deep Learning*.

Applying the predictive modelling power of machine learning to graphs is a natural advance. But conventional techniques such as logistical regression or tree-based algorithms are not immediately compatible with the graph structure, as data instances are expected to be independent and identically distributed. Some attempts to overcome this obstacle by extracting representative features from the graph, like kernel functions (VISHWANATHAN *et al.*, 2010) or graph summary statistics (BHAGAT; CORMODE; MUTHUKRISHNAN, 2011) were presented. *Network embedding* is another class of techniques that aims to model graph data by encoding (projecting) algebraic representation of nodes into vectors from a low-dimensional space in which

geometric properties (*e.g.* distance) represent graph edges (PEROZZI; AL-RFOU; SKIENA, 2014). The aforementioned approaches have downsides: they are dependent on a computationally expensive pre-processing phase and do not generalize if the graph is slightly changed, *e.g.* by adding a new node to it.

Some major limitations of these techniques may be pointed out: the large number of parameters needed to represent network embeddings (proportional to the number of nodes of the graph), the lack of generality of usage, as they cannot generate representations for nodes that were not seen during the training stage. Another limitation is the incapability to use node and edge features of these algorithms, as in real-world applications nodes can be described by a collection of features (node features) as well as the edges (edge features).

Graph Neural Networks (GNNs) (GORI; MONFARDINI; SCARSELLI, 2005) came as a new deep learning approach for graphs. This class of algorithms learns a *computational graph* as a mechanism to share information between nodes of a neighbourhood. Graph nodes are represented as an aggregation of feature information from their neighbours in neural network layers with parameters learnt with back-propagation. GNNs differ from traditional machine learning graph frameworks in the sense that they can process graphs with arbitrary size and topological structure, regardless of node ordering.

During the last decade, different models of GNNs were presented. Taxonomies for understanding such models (WU *et al.*, 2020) organize three main types of tasks for machine learning on graphs: node-level (related to predicting a target feature about the node), edge-level tasks (learning some target feature about the edge or the existence of a link between a pair of nodes) and graph-level (classifying an entire graph by obtaining a compact representation of it). GNNs have also been applied in different learning strategies, depending on the specific objective or data availability:

- **Supervised learning** when the entire graph has the target features to be learnt. It can be applied, *e.g.* for node classification;
- **Unsupervised learning** when there is no information about a target value for the nodes. In this case, the algorithm learns representations for subgraphs or the entire graph;
- **Semi-supervised learning** when only a fraction of the nodes have label information and the model must learn from the labeled nodes and extrapolate the learning to the unlabeled ones;

Having data not completely labeled is a common scenario due to the difficulties and cost to produce data for modelling. Semi-supervised learning has two main approaches to work with such data:

- **Inductive learning** uses available labeled data to produce a model that fills the missing labels;
- **Transductive learning** propagates the labeled data for the unlabeled without generating a classifier.

1.2 Applications

Since the development of GNNs many different applications to problems have been proposed, to which chapter Chapter 3 is dedicated. To name some of these areas:

- **Topic Modelling**: discovering semantic topics from collections of texts (YANG *et al.*, 2020), (ZHOU; HU; WANG, 2020);
- **Computer Vision**: parsing of natural language as text into images (JOHNSON; GUPTA; FEI-FEI, 2018); parsing an image into its representation as a graph of elements (objects of the scene) (XU *et al.*, 2017), (WANG *et al.*, 2019b);
- **Natural Language Processing**: text classification (KIPF; WELLING, 2016a), (HAMILTON; YING; LESKOVEC, 2017), (YAO; MAO; LUO, 2019);
- **Dynamic Systems**: prediction of traffic dynamics (ZHANG *et al.*, 2018); taxi demand prediction (YAO *et al.*, 2018);
- **Recommender Systems**: link prediction (BERG; KIPF; WELLING, 2017) (YING *et al.*, 2018);
- **Others**: Physics (SANCHEZ-GONZALEZ *et al.*, 2020); Biochemistry (LI *et al.*, 2021).

1.3 Objectives and Methods

This master's dissertation presents a detailed study of Graph Neural Networks and their applications. Topic Modelling (the discovery of semantic information from text collections by generative probabilistic models in *Information Retrieval* and *Text Mining*) and text classification are areas with increasing relevance due to the success of new models based on pre-trained parameters. This dissertation proposes text classification model based on propagation on bi-partite graphs and graph neural networks and node features given by a topic model. The objectives of this work are as follows:

- Summarise the knowledge on Graph Neural Networks to serve as a theoretical reference for the field;
- Propose a new text classification using GNNs on bi-partite graphs;

- Compare the proposed model against other well-established techniques.

The modelling method used was applying GNNs for node classification tasks on benchmark text datasets converted into a graph representation. Numeric evaluation of the model performance allows discussion on it's efficiency and on opportunities for future improvement.

1.4 Contributions

The contributions of this master's dissertation are:

- Description and summarization of GNNs;
- Analysis of the use of GNNs on text classification;
- A text classification model using Graph Neural Networks based on bi-partite graphs.

1.5 Structure of this Work

- Chapter 2 - **Background**: Graphs, Neural Networks and Graph Neural Networks concepts are presented;
- Chapter 3 **Variants and Applications**: Presentation of main architectures of GNNs and their applications in machine learning;
- Chapter 4 **Topic Modelling**: Presentation of Topic Modelling problem with prior state-of-the-art techniques and GNN-based models;
- Chapter 5 **Proposal and Results**: Presentation of methods, experimental setup and strategy for evaluating their results;
- Chapter 6 **Conclusion**: Final conclusions and future opportunities are stated.

CHAPTER

2

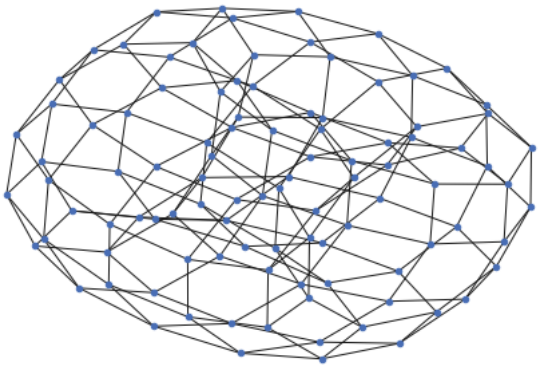
BACKGROUND

2.1 Graphs

A graph can be represented as a data structure, denoted by $G = (V, E)$ in which a node $v_i \in V$ can be connected to a neighbour v_j through an edge $e_{ij} \in E$. The number of vertices of a graph is denoted by $|V|$ and the set of neighbours of an node v_i is denoted by $\mathcal{N}(v_i)$. A very convenient way to describe a graph is by its *adjacency matrix* $\mathbf{A}$, where:

$$A_{ij} = \begin{cases} 1 \text{ if } e_{ij} \in E \\ 0 \text{ if } e_{ij} \notin E \end{cases} \tag{2.1}$$

Figure 1 – Visual representation of a regular square lattice undirected graph



Source: Elaborated by the author.

If $A_{i,j} = A_{j,i}$, for any pair (i, j) the graph is called undirected (directed otherwise) and in this case $\mathbf{A}$ is a symmetric matrix. Directed edges are graphically represented as arrows with points representing the direction of the connection, while undirected edges are represented as line segments linking the connected nodes. When all existing edges have the same intensity, like in equation 2.1 the graph is called unweighted. Otherwise, it is called a weighted graph, with the

graph weight matrix $\mathbf{W}$ defined by the weights (intensity) $W_{i,j}$ of the edges $e_{i,j}$. The number of connections of a node in a graph is called degree k and can be calculated using the adjacency matrix as:

$$k_{in}(v_i) = \sum_j (A_{ji}) \tag{2.2}$$

$$k_{out}(v_i) = \sum_j (A_{ij}) \tag{2.3}$$

where $k_{in}(v)$ and $= k_{out}(v)$ represent respectively the number of directed connections pointing to the edge and coming from it. Note that if the graph is undirected, $k_{in}(v) = k_{out}(v)$. The *degree matrix* $\mathbf{D}$ is defined as a diagonal matrix with the degree of the nodes: $\mathbf{D} = \text{diag}(k_{v_1}, ...k_{v_n})$. The *graph Laplacian* $\mathscr{L}$ is a matrix defined as the subtraction of the degree and the adjacency matrices.

$$\mathscr{L} = \mathbf{D} - \mathbf{A} \tag{2.4}$$

$$\mathscr{L}_{ij} = \begin{cases} k_{v_i} \text{ if } i = j \\ -1 \text{ if } i \neq j \text{ and } v_i \in \mathscr{N}(v_j) \\ 0 \text{ otherwise} \end{cases} \tag{2.5}$$

The Laplacian matrix is useful to represent the spectral decomposition of $\mathbf{A}$ for undirected graphs. It can be demonstrated that $\mathscr{L}$ can be written in terms of the matrix $\mathbf{U}$ in which rows represent the eigenvectors of $\mathbf{A}$:

$$\mathscr{L} = \mathbf{U}\Lambda\mathbf{U}^{-1}, \Lambda = \text{diag}([\lambda_0, \lambda_1, ...\lambda_n]), \tag{2.6}$$

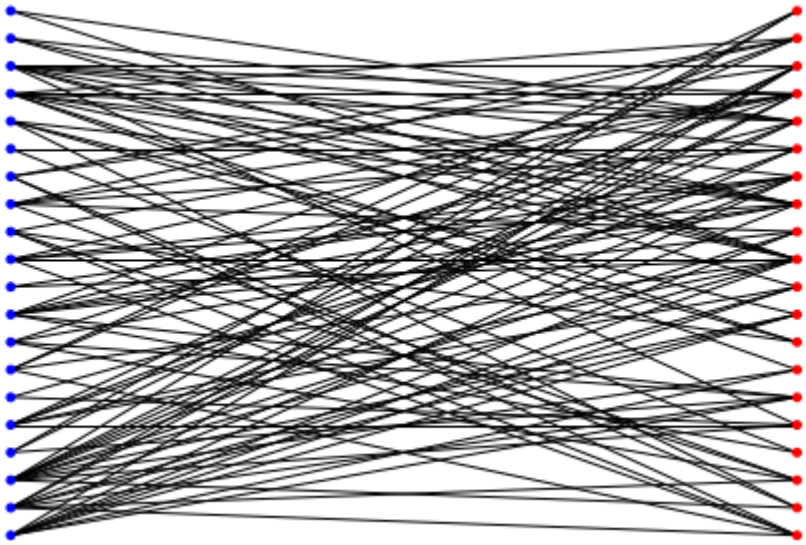
where Λ is a diagonal matrix composed of eigenvalues of $\mathbf{A}$.

Graphs can also be categorized concerning their edge type, denoted by τ, extending the edge notation to $e_{i,\tau,j} \in E$ and defining for each present edge type a separate adjacency matrix $\mathbf{A}_\tau$ for what is called multi-relational graphs. There are two main sets of multi-relational graphs:

- **Heterogeneous graphs:** where there are different disjoint sets of nodes separated by type ($V = V_1 \cup V_2 \cup ... \cup V_k$). In such graphs edge can carry implicit rules, linking only nodes of same set or different sets (*multipartite graphs*);

- **Multiplex graphs:** where the graph is composed of k layers. For each layer edges represent a type of *intra-layer* relation, although *inter-layer* relations are permitted.

Figure 2 – Bipartite graph with randomly connected nodes

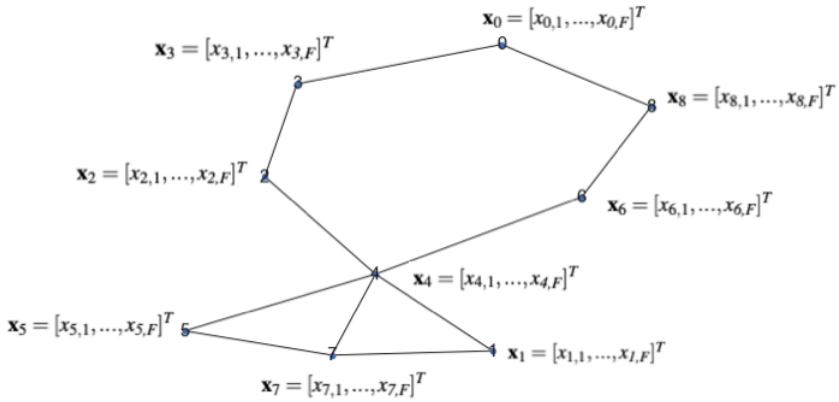


Source: Elaborated by the author.

2.2 Graph Semantic Information

A node v_i may have associated with it a number F of descriptive variables, called *features*, represented by the feature vector $\mathbf{x}_i = [x_1, ..., x_F]^T$. The set of all features nodes is represented by the feature matrix $\mathbf{X} \in \mathbb{R}^{F \times N}$. Similarly, a graph may also have C edge attributes, with $\mathbf{x}^e_{ij} \in \mathbb{R}^C$ representing the feature edge vector of an edge between the pair of edges v_i and v_j. The edge feature matrix is given by $\mathbf{X}^e \in \mathbb{R}^{C \times |V|}$.

Figure 3 – Visual representation of graph node features



Source: Elaborated by the author.

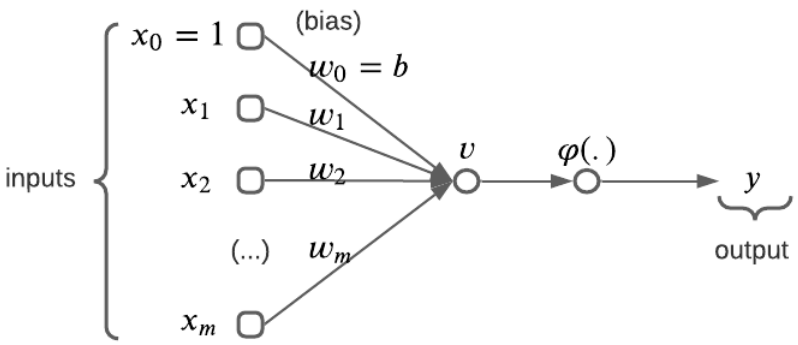
The nodes and edge features of a graph may represent any sort of descriptive data about the dataset elements and their relationships. In predictive modelling, generally, the purpose is to learn one of the features as a function of the remaining ones. Many algorithms of machine learning exist to estimate such functions with multiple application scenarios. For instance: in social media platforms, users may be represented as nodes in a graph and each node may have a set of attributes (age, country, profession, etc.) as node features.

2.3 Neural Networks and Deep Learning

Neural Networks are a class of machine learning algorithms that took inspiration from neuron connections in the nervous system to build artificial neurons. The network of artificial neurons can learn linear separable functions when fed with sufficient amount of input training data (MCCULLOCH; PITTS, 1943).

Rosenblatt (1958) developed an artificial neuron model called *Perceptron* in which input data can be adjusted by weights (called *synaptic weights*) during training. The different input signals (added to a bias sign for generalization) are linearly combined with the weights and pass through a non-linear function that serves as a hard limiter (threshold) to calculate the perceptron's output.

Figure 4 – Rosenblatt's Perceptron



Source: Elaborated by the author.

In Figure 4 the linear projection v of the input signals is given by

$$v = \sum_{i=1}^{m} w_i x_i + b = \sum_{i=0}^{m} w_i x_i \tag{2.7}$$

which can be re-written in the compact matrix form

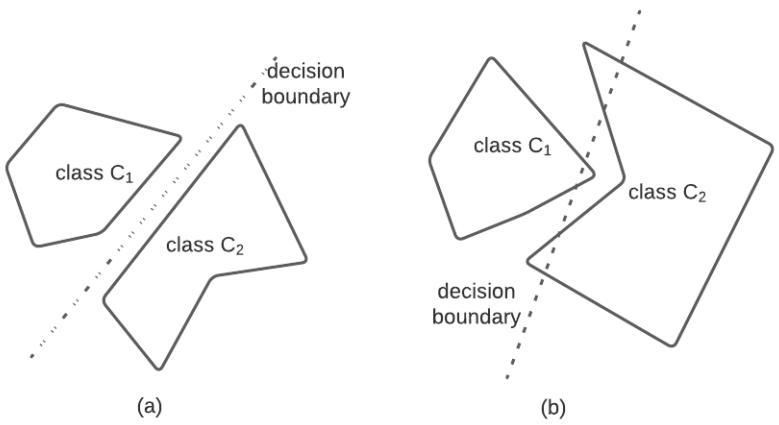
$$v = \mathbf{w}^T \mathbf{x} \tag{2.8}$$

where **w** and **x** are the weight and input vector, respectively. The hard limiter $\varphi(.)$ is a threshold parameter:

$$\varphi(.) = \begin{cases} 1 \text{ if } \upsilon \leq \text{threshold} \\ 0 \text{ if } \upsilon > \text{threshold} \end{cases} \tag{2.9}$$

Equations 2.8 and 2.9 describe a cross-section in a hyper-plane of the input variables, granting that linear-separable functions can be modelled successfully by the perceptron, lacking the adaptability to model non-linear functions, as shown in Figure 5. The training process for the perceptron initializes the values of **w** with 0 or small random values and, for each data sample, calculates the linear projection output y and updates the weight values interatively by subtracting y from the ground-truth value multiplied by a constant until convergence is achieved.

Figure 5 – Pair of linear separable patterns (a) and pair of non-linear separable patterns (b)



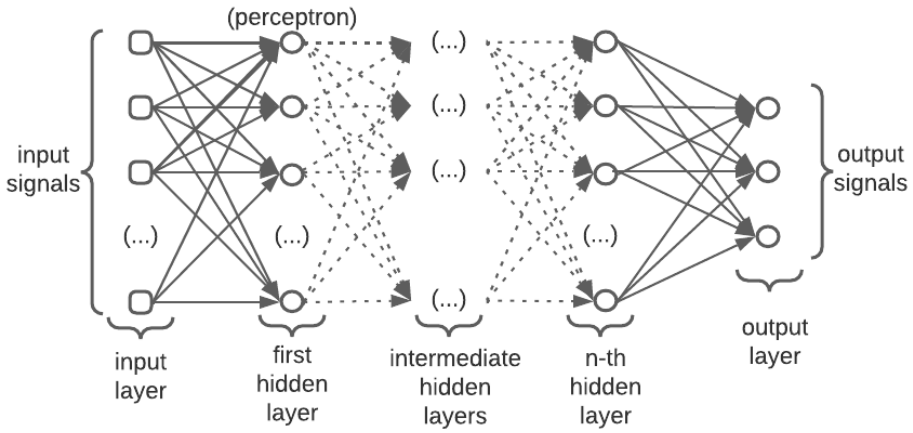
Source: Adapted from Haykin (2010).

Even though a single perceptron (also known as *singe-layer neural network*) has guaranteed convergence on linear-separable functions, it fails when representing non-linear functions. One famous example of this limitation is the XOR operator.

Non-linear functions can be modelled with the use of more than one perceptron acting together, where the result of the linear projection of one perceptron is passed through a non-linear differentiable function called *activation function*. The resulting model is called *multi-layer perceptron* (MLP) . The MLP model has one or more hidden layers of perceptrons (called nodes of the neural network) with a high degree of connectivity between the neighbour layers.

The patterns in input data are extracted by the hidden layers of the neural network of an MLP during the learning process. Training an MLP associates the input data and the *feature space* as each layer of perceptrons performs linear projections and non-linear transformations on output signals of the previous layer.

Figure 6 – Multi-layer perceptron with n fully connected hidden layers



Source: Adapted from Haykin (2010).

Having non-linear differentiable activation functions in the MLP is important for the training process which uses the *Back-propagation* algorithm (RUMELHART; HINTON; WILLIAMS, 1986) to calculate the optimal solution for $\mathbf{w}$ minimizing the error of a *loss function* provided with the labeled data via gradient descent. The development of the Back-propagation algorithm made artificial neural networks computationally efficient and represented a landmark in the research of neural network models with larges amounts of hidden layers, known as *Deep Learning*.

In order to find the *optimum* solution for all weights of the neural network, the Back-propagation algorithm computes the error of each synaptic layer by Chain Rule of Calculus starting from the *error signal* on n-th (output) layer. For the j neuron, the error signal produces is given by the difference of the ground-truth value and the output signal. Having $\mathbf{d}(n)$ defined as the ground-truth vector, the error signal is given by:

$$\begin{aligned} e_j(n) &= d_j(n) - y_j(n) \\ &= d_j(n) - \varphi_j(\upsilon_j(n)) \end{aligned} \tag{2.10}$$

The *total instantaneous error energy* of the network, for all N training examples, is calculated by summing the errors of the set of neurons from the output layer (denoted by C):

$$\begin{aligned} \mathscr{E}(N) &= \sum_{j \in C} \mathscr{E}_j(n) \\ &= \frac{1}{2} \sum_{j \in C} e_j^2(n) \end{aligned} \tag{2.11}$$

After the calculation of the errors, the correction of the synaptic weight $w_{ij}(n)$ (the i-th neuron of the j-th layer calculated at the output layer) is proportional to the partial derivative of

the total instantaneous error energy with respect to the synaptic weight.

$$\frac{\partial \mathscr{E}(n)}{\partial w_{ij}(n)} = \frac{\partial \mathscr{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_i(n)} \frac{\partial y_j(n)}{\partial \upsilon_j(n)} \frac{\partial \upsilon_j(n)}{\partial w_{ij}(n)} \tag{2.12}$$

It can be shown that the correction factor applied to $w_{ij}(n)$ is given by:

$$\Delta w_{ij}(n) = -\eta \frac{\partial \mathscr{E}(n)}{\partial w_{ij}(n)} \tag{2.13}$$

The hyperparameter η is called the *learning rate* and it represents how fast happens the iterative mapping of the weight space in search of the optimal solution. The minus sign asserts that the vector in weight space points to the opposite direction of the gradient of $\mathscr{E}(n)$ (*gradient descent*). The training process initiates with all synaptic weight values with samples from an uniform distribution. When all *N* examples of the training dataset are passed through the network it is said that one *epoch* has passed. The network computed the output signals for all the examples presented (*forward computation*) and then the correction of the synaptic weights happen (*backward computation*). The training process continues iterating as many epochs as needed for reaching convergence. The neural network may also be trained with *stochastic gradient descent*, where samples from the training dataset are presented in batches making the training process less expensive computationally-wise.

Regarding the differentiable, non-linear *activation function* $\varphi(.)$ of the neurons of the MLP, there are typical choices of functions that are presented in Table 1. Each function presents hyperparameters that can be adjusted to fit better different scenarios of pattern detection.

Table 1 – List of common activation functions for MLP

Function Name	**Expression**
Sigmoid	$\varphi_j(\upsilon_j(n)) = \frac{1}{1+exp(-a\upsilon_j(n))}, a > 0$
Hyperbolic Tangent	$\varphi_j(\upsilon_j(n)) = a\tanh(b\upsilon_j(n)), a > 0, b > 0$
Parametric Rectified Linear Unit (ReLU)	$\varphi_j(\upsilon_j(n)) = \begin{cases} \upsilon_j(n) \text{ if } \upsilon_j(n) > 0 \\ a\upsilon_j(n) \text{ otherwise} \end{cases}, a > 0$
Swish	$\varphi_j(\upsilon_j(n)) = \frac{\upsilon_j(n)}{1+exp(-a\upsilon_j(n))}, a > 0$

Source: Research data.

In machine learning the ability of a model to generalize predictions for new data is very appreciated. Good performance measures on training data followed by bad performance on new data often indicate overfitting, when the model adapts to the training data noise. With neural networks, it is avoided with *regularization* techniques that avoid overfitting by penalizing variations in weights during training. A regularization term is added to the loss value. It may be proportional to the sum of the absolute values of the synaptic weights (L1 regularization) or

proportional to the sum of the square values of the synaptic weights (L2 regularization, also called *weight decay*).

$$\text{loss}_{REG} = \begin{cases} \text{loss} + \lambda_{L1} \sum_{i=1}^{num.neurons} \sum_{j=1}^{num.layers} |w_{ij}| \text{ , L1 regularization} \\ \text{loss} + \lambda_{L2} \sum_{i=1}^{num.neurons} \sum_{j=1}^{num.layers} w_{ij}^2 \text{ , L2 regularization} \end{cases} \tag{2.14}$$

Other form of regularization is avoiding small sets of nodes to become specially relevant on training. This technique is called *Dropout*. It sets a probability $p_{dropout}$ of a random perceptron not to be used during the training stage.

2.4 Prior Approaches for Machine Learning with Graphs

Applying machine learning techniques to graph data is a complex task due to the fact that most of well-known algorithms are valid under the statistical hypothesis that data instances are independent, identically distributed (iid) random variables. This assumption does not hold for graph data because nodes are inherently associated amongst themselves. Another difficulty is the fact that machine learning algorithms (*e.g.*, random forest, logistic regression, SVM) and deep learning algorithms rely on regular grid structured input data like tables and images, which graphs cannot produce because nodes have different adjacency regions and therefore might have different representations.

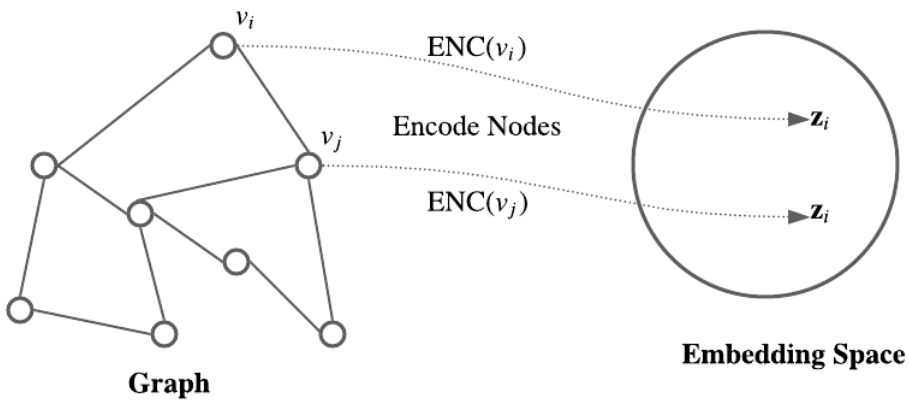
Using the adjacency matrix $\mathbf{A}$ as model input is not possible because even the *iid* hypothesis would suggest the possibility of permutation of elements rows of $\mathbf{A}$, deforming the original structure of the graph. For this reason, it is said that such models are not permutation invariant. A function $f(\mathbf{A})$ is *Permutation invariant* if it does not depend on arbitrary ordering of elements of $\mathbf{A}$. *Permutation equivalence* is the property that the output of f when acting on a permutation of $\mathbf{A}$ is equal to the output of permuting f acting on $\mathbf{A}$. If $\mathbf{P}$ is the permutation matrix, in mathematical terms we have:

$$f(\mathbf{PAP}^T) = f(\mathbf{A}) \text{ (permutation invariance)} \tag{2.15}$$

$$f(\mathbf{PAP}^T) = \mathbf{P}f(\mathbf{A}) \text{ (permutation equivalence)} \tag{2.16}$$

Although there are attempts to use graph data for machine learning purposes focused on mapping the representation of nodes of the graph in a low-dimensional space in such a way that similar nodes in the original graph space are given similar representations in the *embedding space*, the reason why these methods are called *node embedding methods*, which have their main problem represented in Figure 7.

Figure 7 – Node embedding problem



Source: Adapted from Hamilton (2020).

The *encoder* function (ENC) maps each node $v_i \in V$ to a vector $\mathbf{z}_i \in \mathbb{R}^d$. A common practice is the process of mapping nodes by lookup over the node IDs (*shallow embedding*). The *decoder* (DEC) function reconstructs graph representations from the node embeddings. Standard practice is to have pair-wise decoders that predict the similarity between node pairs. The similarity measure $\mathbf{S}$ that can be generically defined by:

$$DEC(ENC(v_i), ENC(v_j)) = DEC(\mathbf{z}_i, \mathbf{z}_j) \approx \mathbf{S}[v_i, v_j] \tag{2.17}$$

This operation is possible by optimizing the empirical reconstruction loss function using the training dataset $\mathscr{D}$, given by:

$$\text{loss} = \sum_{v_i, v_j \in \mathscr{D}} (DEC(\mathbf{z}_i, \mathbf{z}_j), \mathbf{S}[v_i, v_j]), \tag{2.18}$$

where the loss function $l : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, measures the discrepancy between the similarity values of the embedding space and the estimated values by the decoder.

There are a handful of shallow embedding methods and they differ on the approach to produce the low-dimensional embedding space. Some methods optimize the loss function by factorization algorithms acting on a generalization of the adjacency matrix (such as Lap Eigenmaps (BELKIN; NIYOGI, 2001), Graph Factorization (AHMED *et al.*, 2013), GraRep (CAO; LU; XU, 2015) and HOPE (OU *et al.*, 2016)). Other methods, like DeepWalk (PEROZZI; AL-RFOU; SKIENA, 2014) and node2vec (GROVER; LESKOVEC, 2016), use statistics of random walks to compute $p_{G,T}(v_i|v_j)$ the probability of visiting node v_j starting from node v_i after T steps on the graph G. Shallow embedding methods are summarised in Table 2.

Table 2 – Summary of shallow embedding methods

Method	Decoder	Similarity Measure	Loss Function
Lap Eigenmaps	$\|\mathbf{z}_i - \mathbf{z}_j\|_2^2$	General	$DEC(\mathbf{z}_i, \mathbf{z}_j) \cdot \mathbf{S}[v_i, v_j]$
Graph Factorization	$\mathbf{z}_i^T \mathbf{z}_j$	$\mathbf{A}[v_i, v_j]$	$\|DEC(\mathbf{z}_i, \mathbf{z}_j) \cdot \mathbf{S}[v_i, v_j]\|_2^2$
GraRep	$\mathbf{z}_i^T \mathbf{z}_j$	$\mathbf{A}[v_i, v_j], ..., \mathbf{A}^k[v_i, v_j]$	$\|DEC(\mathbf{z}_i, \mathbf{z}_j) \cdot \mathbf{S}[v_i, v_j]\|_2^2$
HOPE	$\mathbf{z}_i^T \mathbf{z}_j$	General	$\|DEC(\mathbf{z}_i, \mathbf{z}_j) \cdot \mathbf{S}[v_i, v_j]\|_2^2$
DeepWalk	$\frac{exp(\mathbf{z}_i^T \mathbf{z}_j)}{\sum_{k \in V} exp(\mathbf{z}_i^T \mathbf{z}_j)}$	$p_{G,T}(v_i \| v_j)$	$-\mathbf{S}[v_i, v_j] log(DEC(\mathbf{z}_i, \mathbf{z}_j))$
node2vec	$\frac{exp(\mathbf{z}_i^T \mathbf{z}_j)}{\sum_{k \in V} exp(\mathbf{z}_i^T \mathbf{z}_j)}$	$p_{G,T}(v_i \| v_j)$ (biased)	$-\mathbf{S}[v_i, v_j] log(DEC(\mathbf{z}_i, \mathbf{z}_j))$

Source: Adapted from Hamilton (2020).

Some limitations can be pointed from shallow embedding methods:

- Parameters are not shared between nodes in the encode (one embedding per node). This leads to high computational complexity ($\mathcal{O}(N = |V|)$).
- Node features are not considered.
- Such methods cannot generalize to produce embeddings for nodes that were not presented during the training stage. This represents a restriction from using the methods on *inductive learning*, for what they are classified as *inherently transductive*.

2.5 Graph Neural Networks

Graph Neural Networks extend the neural networks mechanisms (MLP) to encode graph information. In this work, the formalism for undirected graphs will be described. The generic GNN model consists of representing a node (and its features) using information from its neighbourhood. Such representation can be used to produce predictions about the node. The seminal work (GORI; MONFARDINI; SCARSELLI, 2005) later revised (SCARSELLI *et al.*, 2008) carried the formalism of recurrent neural networks to calculate the state of a node v_i:

$$h_i^{(k+1)} = f(\mathbf{x}_i^{(k)}, \mathbf{x}_j^{(k)}) | j \in \mathcal{N}(v_i) \tag{2.19}$$

where $h_i^{(k+1)}$ is the k-hop representation of node v_i (also called *hidden embedding*), written as function the node features of v_i and its k-hop neighbourhood. The function f represents a neural network adjusted by *Back-propagation* using examples from the training set (supervised and semi-supervised learning requires label feature to be used in the cost function optimization). It is also possible to have an entire graph embedded h_G, usually for graph-focused learning purposes (*e.g.* graph classification, community detection).

The basic mechanism that operates in GNN is called *Neural Message Passing* (GILMER *et al.*, 2017), which is composed of three functions: MESSAGE, AGGREGATION and UPDATE. The k superscript in 2.20 represents the k-th iteration of the message passing mechanism.

MESSAGE computes the node embedding $m_i^{(k)}$ using nodes and edge features

$$m_{ij}^{(k)} = M(h_i^{(k)}, h_i^{(k)}, \mathbf{x}_i^{e,(k)}) | j \in \mathcal{N}(v_i) \tag{2.20}$$

where the message function M is valid in the adjacency region of node v_i and may have different shapes, like simply copying the message values from neighbours, normalizing the message values for the adjacency region or even applying attention (VASWANI *et al.*, 2017) to the messages in order to weight the importance of each neighbour:

$$m_{i,j}^{(k)} = \begin{cases} h_j^{(k)} \text{ (neighbourhood copy)} \\ \frac{1}{c_{i,j}} h_j^{(k)} \text{ (structurally normalized neighbourhood copy)} \\ \alpha(h_i^{(k)}, h_j^{(k)}).h_j^{(k)} \text{ (attention)} \end{cases} \tag{2.21}$$

The *AGGREGATE* function computes a fixed-length representation of the neighbourhood of the embedded node (regardless of its size):

$$\hat{m}_i^{(k)} = \oplus_{j \in \mathcal{N}(v_i)} m_{ij}^{(k)} \tag{2.22}$$

where $\oplus$ is a permutation invariant transformation. Some options for such transformations are the sum or the maximum value of neighbour's messages. For cases when there are hubs (nodes with large amounts of connections compared to others in the graph) may decrease the quality of descriptive power of the GNN. Neighbourhood normalization and symmetric neighbourhood normalization (KIPF; WELLING, 2016b) correct this effect.

$$\hat{m}_i^{(k)} = \begin{cases} \sum_{j \in \mathcal{N}(v_i)} m_{i,j}^{(k)} \text{ (sum)} \\ \max_{\mathcal{N}(v_i)} m_{i,j}^{(k)} \text{ (maximum)} \\ \sum_{j \in \mathcal{N}(v_i)} \frac{h_i^{(k)}}{|\mathcal{N}(v_i)|} \text{ (neighbourhood normalization)} \\ \sum_{j \in \mathcal{N}(v_i)} \frac{h_i^{(k)}}{\sqrt{|\mathcal{N}(v_i)||\mathcal{N}(v_j)|}} \text{ (symmetric neighbourhood normalization)} \end{cases} \tag{2.23}$$

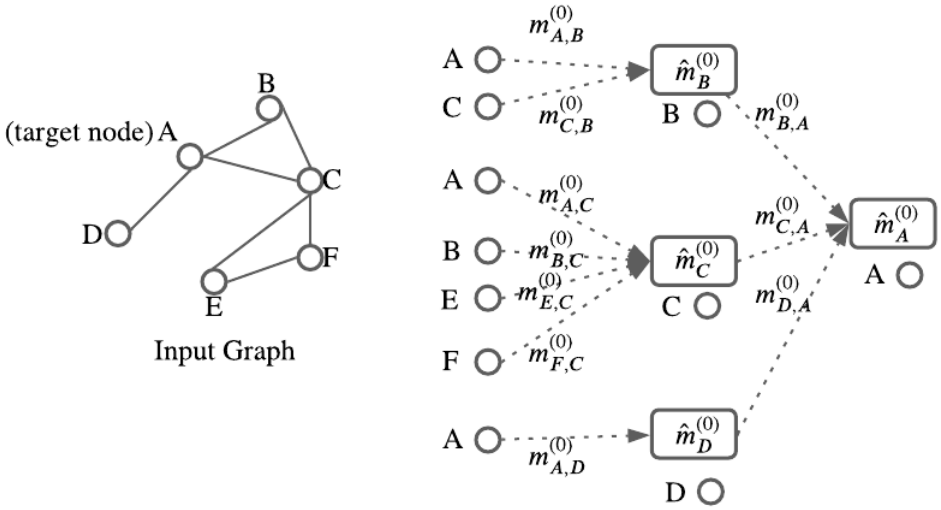
Regarding the *AGGREGATE* function, it was proven (ZAHEER *et al.*, 2017) that it is a *universal set approximator* if it satisfies equation 2.15 (permutation invariance), mapping a set of embeddings to a single embedding, for a set of learnable parameters θ:

$$\hat{m}_i^{(k)} = \text{MLP}_\theta\Big(\sum_{j \in \mathcal{N}(v_i)} \text{MLP}_\varphi h_j^{(k)}\Big) \tag{2.24}$$

Once the messages are aggregated, the *UPDATE* function computes new node embeddings using aggregated messages and the old embeddings:

$$h_i^{(k+1)} = \varphi(h_i^{(k)}, \hat{m}_i^{(k)}) \tag{2.25}$$

Figure 8 – Neural Message Passing Diagram



Source: Adapted from Hamilton (2020).

with different models of GNNs applying particular definitions of the *UPDATE* function.

The overall mechanism of Neural Message Passing is illustrated in Figure 8, where the local neighbourhood of node A passes messages to embed this node based on their respective adjacency region (reason for the $(k=0)$ mark on messages).

The process of neural message passing through k iterations results in the final embedding for each node of the graph. In MLP terms node embeddings are given by:

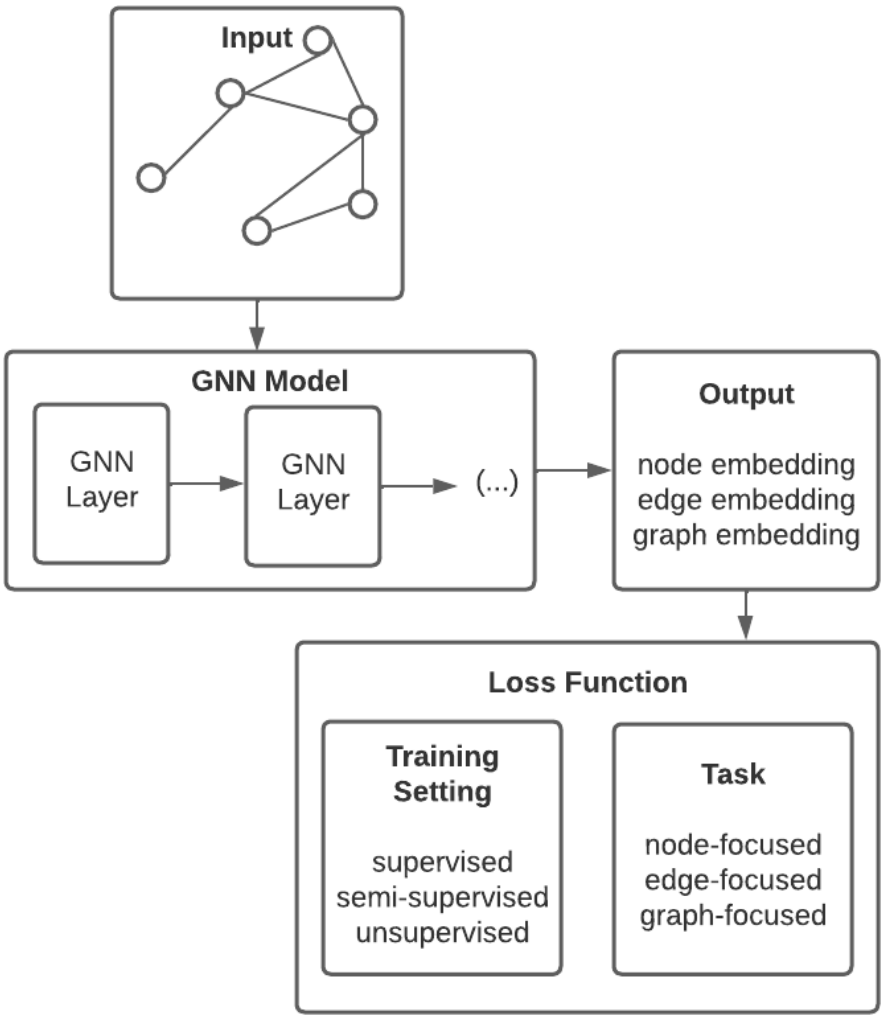
$$h_i^{(k+1)} = \varphi[\mathbf{W}_{self}^{(k+1)} h_i^{(k)} + \mathbf{W}_{\mathcal{N}}^{(k+1)} \sum_{j \in \mathcal{N}(v_i)} h_j^{(k)} + b^{(k+1)}] \tag{2.26}$$

where φ is an element-wise non-linearity, the weights matrices $\mathbf{W}_{self}$ and $\mathbf{W}_{\mathcal{N}}$ are learnable parameters and b is the bias term. Equation 2.26 can be simplified by adding self loops on the *AGGGREGATE* phase of message passing, which is equivalent to computing *UPDATE* in the same phase by sharing parameters between the weight matrices.

$$h_i^{(k+1)} = \varphi[(\mathbf{A}+\mathbf{I})\mathbf{W}^{(k+1)} \sum_{j \in \mathcal{N}(v_i) \cup v_i} h_j^{(k)} + b^{(k+1)}] \tag{2.27}$$

Equations 2.7 and 2.26 are similar as GNNs and MLP are both based on layers of linear projections of input signals followed by an element-wise nonlinear transformation. This framework can effectively represent nodes, edges and even entire graphs. Edge representation learning will not be regarded in detail in this master's dissertation. The basic pipeline for GNN modelling is expressed in Figure 9.

Figure 9 – Basic pipeline for GNN modelling



Source: Adapted from Zhou *et al.* (2020).

2.6 Summary

Choosing adequate data structures when solving specific tasks can heavily influence the solution's effectiveness. When data is presented in singular shapes such as with graphs, algorithms must either preprocess the data (losing valuable information in this costly stage) or adapt to the graph topology in the case of Graph Neural Networks.

In this chapter, the main concepts of graph algebra and machine learning with graphs were presented. The fundamentals of deep learning and GNNs are established as well as the generic framework for solving different classes of problems in machine learning.

The mathematical formalism for Graph Neural Networks in a general form was stated and will be used to discuss different models from literature in the following chapter.

CHAPTER

3

VARIANTS AND APPLICATIONS

In this chapter, prior related work on GNN models from the literature are reviewed and organized by method type. By no means this work intends to touch all available variants. At the end of the chapter, there is an analysis of GNN applications with perspectives for the field.

3.1 Spectral Methods

Spectral Methods rely on the Fourier transform to generate a convolutional operator in the spectra domain of $\mathbf{A}$ using graph signal processing framework (SHUMAN *et al.*, 2013). The convolution operation defined over finite domains (denoted by the operator $\star$) represents the filtering operation of a function f (the *signal*) by a filter h.

$$(f \star h)(\mathbf{x}) = \int f(\mathbf{y}) h(\mathbf{x}-\mathbf{y}) d\mathbf{y} \tag{3.1}$$

The filter in most cases represents a domain in which the signal propagates (*e.g.* time t). A shift in the signal domain passed by a convolution is equivalent to convolving the signal and then transliding the final result:

$$\Delta f(t) \star g(t) = f(t) \star \Delta g(t) = \Delta (f \star g)(t) \tag{3.2}$$

with $\Delta f(t) = f(t+1) - f(t)$ being called the Laplace operator.

In graph domains, such filtering operations are equivalent to performing a walk on connected edges. This node shift can be achieved by multiplying a vector $\mathbf{f}$ (representing the current node) by the *circulant matrix* $\mathbf{A}_C$, that replaces the adjacency matrix:

$$A_c[i,j] = \begin{cases} 1 \text{ if } j = (i+1)_{\text{mod}N} \\ 0 \text{ otherwise} \end{cases} \tag{3.3}$$

$$(\mathbf{A}_C \mathbf{f})[t] = \mathbf{f}[(t+1)] \tag{3.4}$$

Using the unnormalized Laplacian $\mathscr{L}_C = \mathbf{I} - \mathbf{A}_C$ we have

$$(\mathscr{L}_C \mathbf{f})[t] = \mathbf{f}[(t+1)_{\text{mod}N}] - \mathbf{f}[t] \tag{3.5}$$

showing the similarity between time shifts on the graph signal by Fourier convolutions and the node signal propagation in a graph using multiplications adjacency and Laplacian matrices. Generally, a convolutional filter in a graph can be defined as a matrix operator $\mathbf{Q}_h$ that commutes with $\mathbf{A}$ and $\mathscr{L}$ in the form

$$\mathbf{Q}_h = \alpha_0 \mathbf{I} + \sum_{n=1}^{N} \alpha_n \mathbf{A}^N \tag{3.6}$$

and the features of a node can be convolved by $\mathbf{Q}_h$ resulting in its representation by the N-hop node adjacency region, previously referred as *message*

$$\mathbf{Q}_h \mathbf{X} = \alpha_0 \mathbf{I}\mathbf{X} + \sum_{n=1}^{N} \alpha_n \mathbf{A}^N \mathbf{X} \tag{3.7}$$

In order to represent graph convolutions using Fourier formalism, recalling the eigendecomposition of the Laplacian matrix (equation 2.6), the Fourier transform of the signal $\mathbf{f}$ and its inverse can be written respectively as

$$\begin{aligned} \mathscr{F}(\mathbf{f}) &= \mathbf{U}^T \mathbf{f} \\ \mathscr{F}^{-1}(\mathbf{f}) &= \mathbf{U}^T \mathbf{f} \end{aligned} \tag{3.8}$$

For some filter $\mathbf{h}$, the convolution theorem (MALLAT, 1999) allows the Fourier coefficients to be calculated by a graph convolution using element-wise products:

$$\mathbf{f} \star \mathbf{h} = \mathscr{F}^{-1}(\mathscr{F}(\mathbf{f}) \odot \mathscr{F}(\mathbf{h})) \tag{3.9}$$

3.2 Convolutional GNN Variants

The theoretical generalization of convolutions graphs was used by Bruna *et al.* (2013) to define the *Spectral Network*, which combines multiple convolutional layers and non-linearities, having the filter θ_h given by learnbale parameters of a diagonal matrix corresponding to a meaningful convolution on the graph:

$$\begin{aligned} \mathbf{f} \star \mathbf{h} &= \mathbf{U}(\mathbf{U}^T \mathbf{f} \odot \mathbf{U}^T \mathbf{h}) \\ &= (\mathbf{U}\text{diag}(\theta_h)\mathbf{U}^T)\mathbf{f} \end{aligned} \tag{3.10}$$

where $\theta_h = \mathbf{U}^T \mathbf{h}$ is the spectral domain representation of $\mathbf{h}\text{diag}(\theta_h)$. This operation is not computationally efficient. Also, results in a filter are not spatially localized. As an alternative, by

choosing the spectral filter as N-degree polynomial of the eigenvalues (HAMMOND; VANDERGHEYNST; GRIBONVAL, 2011) of $\mathscr{L}$ the spatial locality is then ensured:

$$\begin{aligned} \mathbf{f} \star \mathbf{h} &= (\mathbf{U} p_N(\Lambda) \mathbf{U}^T)\mathbf{f} \\ &= p_N(\mathscr{L})\mathbf{f} \end{aligned} \tag{3.11}$$

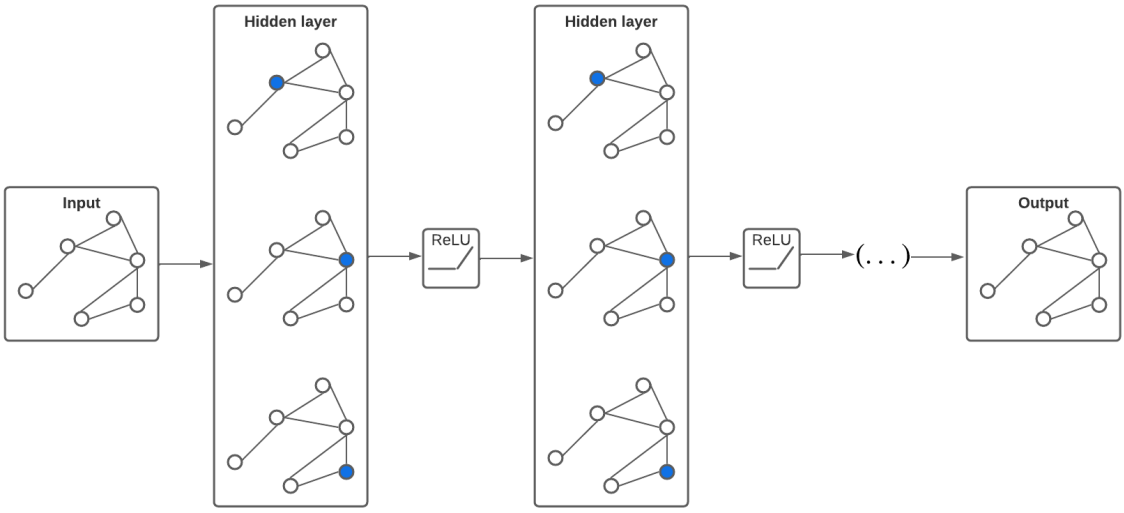
Graph Convolutional Network (GCN) (KIPF; WELLING, 2016a) is one of the most popular GNN architectures. The model uses an approximation for the polynomial solution class similar to equation 3.11 called Chebyshev polynomials, reducing the time complexity of the algorithm. The network is defined as

$$\mathbf{H}^{(k+1)} = \varphi(\tilde{\mathbf{A}}\mathbf{H}^{(k)}\mathbf{W}^{(k+1)}) \tag{3.12}$$

where $\mathbf{W}^{(k+1)}$ is a learnble matrix and the adjacency matrix is replaced by the normalized adjacency matrix $\tilde{\mathbf{A}}$ given by $(\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}(\mathbf{I}+\mathbf{A})(\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}$. The model can be re-written to fit the *message passing* framework (recalling equation 2.26) as

$$\mathbf{H}^{(k+1)} = \varphi(\mathbf{H}^{(k)}\mathbf{W}^{(k+1)}_{self} + \mathbf{A}\mathbf{H}^{(k)}\mathbf{W}^{(k+1)}_{j \in \mathscr{N}(v_i)}) \tag{3.13}$$

Figure 10 – Multi-layer Graph Convolutional Network (GCN) with first-order filters



Source: Adapted from Kipf and Welling (2016a).

Probabilistic Graph Models (PGM) are a class of GNNs that generalizes the graph representation for Markov random fields. Node features $\mathbf{x}$ are mapped to probability distribution $p(\{\mathbf{x}\})$ and node embeddings $\mathbf{z}$ are given by parametric functions Φ and Ψ that carry the formalism of Hilbert space algebra. For $G = (V, E)$:

$$p(\{\mathbf{x}_i\}, \{\mathbf{z}_j\}) \propto \prod_{i \in V} \Phi(\mathbf{x}_i, \mathbf{z}_i) \prod_{(i,j) \in E} \Psi(\mathbf{z}_i, \mathbf{z}_j) \tag{3.14}$$

Using this formalism, the model is built by computing the likelihood of a set of embeddings given its observed features. Such a model is not scalable. Variational inference is employed to produce an approximation q for equation 3.14.

$$p(\{\mathbf{x}_i\}, \{\mathbf{z}_j\}) \approx q(\{\mathbf{z}_i\}) = \prod_{i \in V} q(\mathbf{z}_i) \tag{3.15}$$

Kullback–Leibler (KL) divergence (KULLBACK; LEIBLER, 1951) between true posterior and approximate posterior is used as loss function to be minimized in order to obtain the probabilistic representations of the node embeddings μ. The formal process is out of the scope of this dissertation. Dai, Dai and Song (2016) defines the probabilistic GNN as:

$$\mu_i^{(t+1)} = \varphi(\mathbf{W}_{self}^{(t+1)} \mathbf{x}_i + \mathbf{W}_{\mathcal{N}}^{(t+1)} \sum_{j \in \mathcal{N}(v_i)} \mu_j^{(t)}) \tag{3.16}$$

which is structurally similar to equation 2.19. The weight matrices are updated with an iteractive gradient descent process.

Graph Isomorphism Network (GIN) is a model that leverages the concept of isomorphism, the topological equivalence properties that one graph may have in relation to one another. Two graphs G_1 and G_2 are isomorphic if they share the same structure, differing in node order given by a permutation operation given by

$$\mathbf{P}\mathbf{A}_1\mathbf{P}^T = \mathbf{A}_2 \tag{3.17}$$

The Weisfeiler-Lehman (WL) algorithms for isomorphism test (WEISFEILER; LEMAN, 1968) are a class of heuristic techniques to check for graph isomorphism with polynomial complexity $\mathcal{O}|V|^p)$, avoiding the naive approach of computing all possible permutation matrices in time $\mathcal{O}(|V|!)$. The simplest algorithm, 1-WL assigns each node to a colour $c_i^{(0)}$ and then iteratively reassigns this colour with neighbourhood information aggregation:

$$c_i^{(t+1)} = \text{HASH}(c_i^{(t)}, \{\{c_{j \in \mathcal{N}(v_i)}^{(t)}\}\}) \tag{3.18}$$

Convergence is achieved when colours stop changing and two graphs are possibly isomorphic if they have the same colour histogram (non-isomorphic otherwise). For GNNs,

isomorphism is important because it provides a way to quantify the representational power of a given GNN. Graph Isomorphism Network (GIN) (XU *et al.*, 2018) provides a model with the least amount of parameters that is still as powerful as de WL algorithm, defined the addition of a learnble parameter ε to the *UPDATE* function:

$$h_i^{(k+1)} = \text{MLP}^{(k)}[(1+\varepsilon^{(k)})h_i^{(k)} + \sum_{j \in \mathcal{N}(v_i)} h_j^{(k)}] \tag{3.19}$$

3.3 Spatial Methods

Spatial approaches rely on the graph topology to perform convolution operations. Aggregations are made locally, from one neighbourhood to the other. Such methods formulate convolutions parameters that can be shared with different sizes of neighbourhoods. Their main advantage is the fact that convolution can be computed in batches and shared with different locations and or unseen graph structures, therefore they may be suitable for dealing with large graphs. Unlikely spectral-based methods, spatial methods can handle directed graphs.

GraphSAGE (HAMILTON; YING; LESKOVEC, 2017) is a well-known spatial-based variant of GNNs that reached state-of-the-art performance in node-focused and graph-focused classification tasks (supervised and unsupervised). Feature information is learned through a generalized k-hop neighbourhood process. By having this local approach, node embeddings can be created with less memory, ignoring the rest of the graph when generating a computation graph. Selecting a set of computation graphs (mini-batch sampling) is a way to leverage the model training. GraphSAGE (an acronym for Graph SAmpe and aggreGatE) also samples the neighbourhood nodes during training in order to reduce the cost of computation graph for hubs (nodes with high degree), increasing precision variance as a trade-off. Another contribution of this method is that since generalized neighbourhood sampling is made with a fixed size, the trained model can be used to classify unseen graphs in a inductive set. Different aggregation functions are used for GraphSAGE, for example, the *pooling aggregator* in which each neighbours's vector is independently passed through a fully connected MPL, followed by an element-wise max-polling operator.

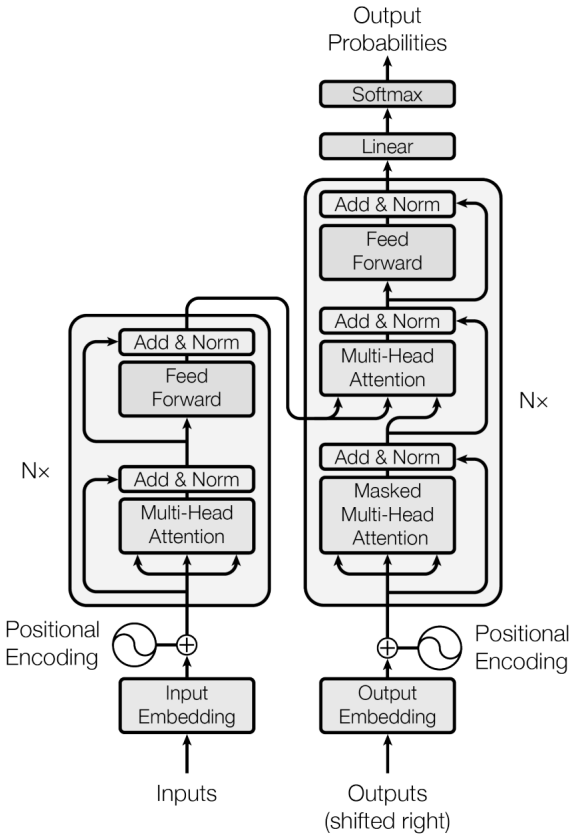
$$h_i^{(k+1)} = \max(\text{MLP}^{(k)}[h_i^{(k)}, h_j^{(k)}]), j \in \mathcal{N}(v_i) \tag{3.20}$$

3.4 Attention-based Methods

The attention mechanism is a technique in deep learning for increasing the relevance of some neural connections in comparison to others. It works by computing the weighted sum of output connections and then assigning a compatibility metric to the output values in order to identify the most significant ones. The use of attention on Encoder-Decoder in the

Transformer model (VASWANI *et al.*, 2017) has successfully solved different classification tasks achieving state-of-the-art performance for text processing tasks. Initially applied to translation, the transformer uses positional encoding and self-attention to generate vector representations of texts regarding the context of phrases. It can be shown that transformer neural networks can be generalized as arbitrary graph representations (DWIVEDI; BRESSON, 2020).

Figure 11 – Transformer Model



Source: Adapted from Vaswani *et al.* (2017).

By using attention weights for neighbours of a node, *Graph Attention Networks* (GAT) (VELIČKOVIĆ *et al.*, 2017) creates a model that expresses the original node features as a product of a *graph attention layer*, its version of the computational graph. Attention weights are computed as matrix products for every node.

$$e_{ij} = \alpha(\mathbf{W}\mathbf{x}_i, \mathbf{W}\mathbf{x}_j) \tag{3.21}$$

and they express the importance of node i to node j. The learnble hyperparameter α is then normalized across all the combinations of nodes (defined by the graph structure):

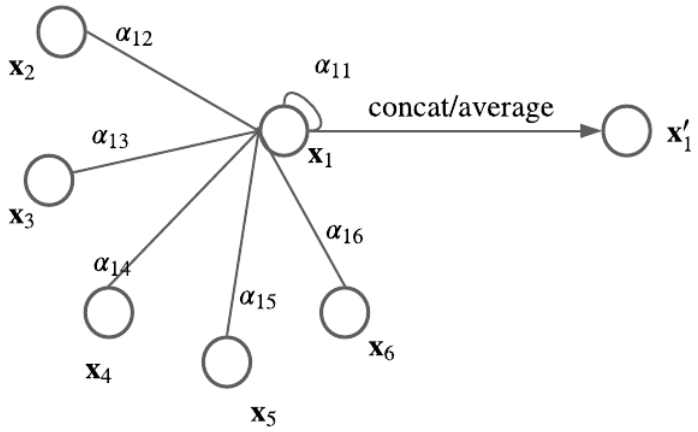
$$\alpha_{ij} = \text{softmax}(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k \in \mathcal{N}_{v_i}} exp(e_{ik})} \tag{3.22}$$

The node features resulting from K graph attention layers are a concatenation (denoted by the $\|$ operator) of all of the intermediary MLP layers (also possible by averaging them):

$$\mathbf{x}'_i = \|_{k=1}^{K} \varphi[\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \mathbf{x}_j] \tag{3.23}$$

GAT achieved state-of-the-art performance in transductive and inductive sets. It is also a low-cost model because the final layers have the same number of parameters, independently from the graph size.

Figure 12 – Graph attention layer



Source: Adapted from Veličković *et al.* (2017).

GATv2 (BRODY; ALON; YAHAV, 2021) expands the calculation of attention to the dynamic weighting of neighbour embeddings increasing the expressiveness of the network. By changing the order of operations of GAT, GATv2 writes equation 3.22 as

$$\mathbf{x}'_i = \alpha_{ij}^k \|_{k=1}^{K} \varphi[\sum_{j \in \mathcal{N}_i} \mathbf{W}^k \mathbf{x}_j] \tag{3.24}$$

Graph Transformer Network (GTN) (YUN *et al.*, 2019) is a network architecture that has shown in experiments to address well the handling of heterogeneous graphs. Prior techniques pre-processed heterogeneous graphs converting them into homogeneous graphs with connections defined by *meta-pahts* (a path consisting of a sequence of relations between different object types, *e.g.* document-word relations in text knowledge graphs). In GTN, the computational graph is called *Graph Transformer (GT) Layer* and it consists of computing a tensor as a product of the intermediary adjacency matrices (called adjacency tensor). The node representation is learned

after stacking (concatenating) k GT layers, forming a *meta-path*:

$$Z = \|_{i=1}^{C} \varphi[\tilde{D}_i^{-1}\tilde{A}_i^k XW] \tag{3.25}$$

where C denotes the number of channels (dimensionality of the output layer). The stacked GT layers are then passed by a CGN. This model was used in different node classification problems, achieving state-of-the-art performance.

3.5 Applications

Node classification problem is defined as learning useful representation for graph nodes in order to correctly classify node i accordingly to its true class or label, denoted by $\hat{y}_i$. Under the *supervised node classification* set, the whole graph is labeled and the training is performed using a fraction of the nodes, to be tested against the remaining portion. Different GNN models have achieved state-of-the-art when applied to this machine learning set. To name a few: GCN (KIPF; WELLING, 2016a), GAT (VELIČKOVIĆ *et al.*, 2017), NPNN (GILMER *et al.*, 2017). Statistically, this problem is expressed by an optimization problem and the solution is found by minimizing the loss function

$$O = \frac{1}{n_l} \sum_{i=1}^{n_l} \text{loss}(\hat{y}_i, y_i) \tag{3.26}$$

where y_i is the class predicted by the algorithm for node i. In classification problems the loss function is commonly chosen to be the *cross-entropy* loss function. The learnble parameters of the GNN are found by having $h^{(k)}$, the last GNN layer passed by a softmax layer that maps the node representations into the output (class) space:

$$\hat{y}_i = \text{softmax}(\mathbf{W} h_i^{(k)}) \tag{3.27}$$

Semi-supervised node classification is defined as learning node labels when not all nodes have node information. This type of problem can be addressed in the *transductive set*, when all nodes are passed to the algorithm, including unlabeled nodes which are then classified. It is also possible to address the problem in the *inductive set*, when the algorithm learns to classify nodes that were not offered during the trainging process. As examples of models used for semi-supervised learning there is CGN (KIPF; WELLING, 2016a), VAE (KIPF; WELLING, 2016b) (transductive set) and GraphSAGE (HAMILTON; YING; LESKOVEC, 2017) (inductive set). The loss function is generally written in terms of the Kullback-Leibler divergence between probability distributions given by equation 3.15.

Link prediction problem is defined as predicting the existence of an edge between two nodes of a graph, generally this task is presented as a *semi-supervised* learning problem. When dealing with heterogeneous bipartite user-term graphs link prediction is often referred to as matrix completion or recommender systems (WU *et al.*, 2022). *Node-based link prediction* methods treat GNNs as *inductive* network embedding methods. Node embeddings are learnt from this representation and pairwise node embeddings are aggregated representing edges. One example of GNN application in this scenario is GAE (KIPF; WELLING, 2016b), which has adjacency and features matrices $\mathbf{A}$ and $\mathbf{X}$ passed through GCN (KIPF; WELLING, 2016a) to get the node representations output $\mathbf{z}_i$ and then passes pairs of nodes (i, j) representations by

the sigmoid function to obtain the link representation, $\hat{A}_{ij}$ interpreted as the probability for the existence of an edge between nodes (i, j):

$$\hat{A}_{ij} = \varphi(\mathbf{z}_i^T \mathbf{z}_j), \text{ where } \mathbf{z}_i = \text{CGN}(\mathbf{X}, \mathbf{A})_i \tag{3.28}$$

The model learns the link representations by minimizing the cross-entropy between the adjacency matrix reconstructed from all $\hat{A}_{ij}$ and the true adjacency matrix:

$$\text{loss} = \sum_{i,j} [-A_{ij} \log(\hat{A}_{ij}) - (1 - A_{ij}) \log(1 - \hat{A}_{ij})] \tag{3.29}$$

Subgraph-based link prediction methods use GNNs to learn from local subgraph representation around target links. One example this type of method is the SEAL (learning from Subgraphs, Embeddings and Attributes for Link prediction) framework (ZHANG; CHEN, 2018). This algorithm first selects target links and extracts subgraphs around them, applying node labeling to mark nodes inside the subgraph. Then the labeled subgraphs are fed into a GNN responsible for learning the graph-level structure features of the subgraph in order to perform the link prediction.

Graph classification problem is defined as extracting graph (or subgraph) features in order to correctly predict its label. It can be applied in both supervised and unsupervised sets, using a large range of models. Computational simulation of molecular structures represented as graphs has become a growing research topic for GNNs in the pharmaceutical industry. As examples in of this type of application (drug discovery), Gaudelet *et al.* (2021) using GIN (XU *et al.*, 2018) and Stokes *et al.* (2020) adapting neural message passing (GILMER *et al.*, 2017).

3.6 Model Interpretability

A common discussion regarding deep learning models is the lack of interpretability concerning the components of the MPL, for which such models receive the "black box" label as a criticism. GNNs inherit this undesired characteristic, which has been targeted by different approaches. Although attention-based GNNs are natural candidates for bringing interpretability for the models by identifying the most relevant connections of a neighbourhood, in recent years there were some efforts to increase the interpretability, specifically for other GNN models addressing the problem from different perspectives. A taxonomy for the available techniques on GNN interpretability (YUAN *et al.*, 2020b) groups them into the instance-level and model-level explanations

Instance-level explanations explore gradients or features to explain the model. Sensitivity Analysis (BALDASSARRE; AZIZPOUR, 2019), Integrated Gradients (SANCHEZ-LENGELING *et al.*, 2020) are examples of using MLP structural gradients to correlate predic-

tions and input values. Methods like GraphLime (HUANG *et al.*, 2022) and RelEx (ZHANG; DEFAZIO; RAMESH, 2021) use classical interpretable models like regression and tree-based algorithms to fit the GNN model. LRP (BALDASSARRE; AZIZPOUR, 2019) and GNN-LRP (SCHNAKE *et al.*, 2020) are examples of techniques called relevance-propagation, in which a rule is defined for the activation redistribution between the neurons of the network. At last, perturbation-based techniques like GNNExplainer (YING *et al.*, 2019) and PGExplainer (LUO *et al.*, 2020) rely on modelling a mask for edges, nodes and features that are combined with the input graph to obtain a new graph representation that contains important input information.

Model-level explanations provide a general, high-level description of the GNN, like pointing out what kind of graph patterns can lead to specific target predictions. XGNN (YUAN *et al.*, 2020a) is an example of this group and its objective is to train graph generators by reinforcement learning, feeding the generated graph to the GNN model. XGNN pipeline allows it to describe graph classification models only.

3.7 Computational Implementations

Graph neural networks are an emerging research area that has contributed with a plethora of models applied in may different scenarios. Different computational libraries were proposed for modelling graph data for GNN using open-source Python language:

- Deep Graph Library (WANG *et al.*, 2019a);
- PyTorch Geometric (FEY; LENSSEN, 2019);
- Spektral (GRATTAROLA; ALIPPI, 2021);
- Graph-Nets (BATTAGLIA *et al.*, 2018);
- PyDGN (BACCIU *et al.*, 2020);
- Alibaba AliGraph (ZHU *et al.*, 2019);
- AWS Daemon (VIRINCHI; SALADI; MONDAL, 2022);

Pytorch Geometric has the special feature of allowing to train models using GPUs and is the most cited engine in recent GNN research.

3.8 Summary

Graph Neural Networks is an emerging field of research that allows using graph-structured data for different deep learning purposes. In this chapter the main GNN variants were discussed as well as research applications in which this approach achieved state-of-the-art performance.

Many domains of knowledge start to be explored with GNN as new model variants are proposed. Next chapter follows an analysis of Topic Modelling, a class of models for Information Retrieval from text data.

CHAPTER

4

TOPIC MODELLING

4.1 Topic Modelling Techniques

Text is a very common format to express real word data, growing in volume ever since digital text storage has become available to companies and consumers. For its unstructured nature, processing text data is a necessary step to retrieve information such as summarisation and classification. There are different approaches to extracting analytical value from text data deriving from branches of research like *Text Mining* and *Information Retrieval* (IR) . Since reviewing the Text Mining knowledge domain is out of the scope of this work, some basic terminology is presented for later reference:

- **word** w is the basic unit of discrete text data extracted from an indexed set (called vocabulary $\mathscr{V}$) with size N. $|\mathscr{V}| = N$.
- **document w** is a portion of written text formed by a sequence of n words: $\mathbf{w} = (w_1, ..., w_n)$
- **corpus** $\mathscr{D}$ is a collection M documents: $\mathscr{D} = \{\mathbf{w}_1, ..., \mathbf{w}_M\}$
- **topic** z is a semantic meaning expressed by a word (or group of words) form the text
- **cluster** is a group of words that share a defined similarity

The discovery of abstract semantic structure (topics) from text using statistical models is called *Topic Modelling* (TM). In these techniques, the order of the words inside the document is neglected (*bag-of-words* assumption). Early relevant TM models, like Baeza-Yates, Ribeiro-Neto *et al.* (1999), consisted in converting the text document into a word frequency vector. *Term frequency-inverse document frequency* (TF-IDF) is a metric used to reduce the importance of words commonly shared across many documents.

$$TF - IDF(w_i, \mathbf{w}) = \frac{\text{count}(w_i \in \mathbf{w})}{N} \log\left(\frac{N}{\text{count}(\mathbf{w} \in D | w_i \in \mathbf{w})}\right) \tag{4.1}$$

The normalized vectors for the entire corpus are then used to provide information about the texts. Successive models targeted dimensionality reduction to generate efficient corpus representation that still contains most of the variance from the original documents. *Latent Semantic Analysis* (LSA) (DEERWESTER *et al.*, 1990) approached this challenge by mapping documents and words to a *latent semantic* space done by a matrix operation called *Singular Value Decomposition* (SVM), with the hypothesis that the semantic similarity from the words of the documents is also represented in the latent space. Despite its lack of statistical foundations, Latet Semantic Indexing (LSI) has been successfully applied to IR in different scenarios, in particular, automatic LSI of texts.

Probabilistic Latent Semantic Indexing (pLSI) (HOFMANN, 1999) has every word in a document as a sample from a mixture model and the semantic topics from the corpus are the mixture components, modeled as random variables. The document is then represented as a list of mixing proportions (probability distributions) for the topics. In the pLSI model, the word w_n and the document $\mathbf{w}$ are conditionally independent given an unobserved topic z according the probability:

$$p(\mathbf{w}, w_n) = p(d) \sum_z p(w_n|z) p(z|\mathbf{w}) \tag{4.2}$$

where d is a multinomial random variable that assumes different values during the training of the model, in which the topic mixtures $p(z|d)$ are learned. Equation 4.2 offers a generative parametric model for words from the corpus given a set of topics.

Latent Dirichlet Allocation (LDA) (BLEI; NG; JORDAN, 2003) is a generative probabilistic model, *i.e.*, randomly generates values from latent variables, for a corpus with the capability to generate probabilities for words from unseen documents. In Topic Modelling, the observable variables are the collection of words in the documents while the latent variables are the topic distributions. LDA provides a model for explaining how to compose documents from these distributions given *a priori* a set of hyper-parameters.

For each document $\mathbf{w}_j$ in a corpus D, N words are chosen using Poisson distribution ($N \sim \text{Poisson}(\xi)$) . A distribution θ is calculated from the Dirichlet distribution with hyper-parameter α. Distributions ϕ_k for topics over the words are calculated with Dirichlet distribution with hyper-parameter β. Then for each word i a topic $z_{j,i}$ is chosen from multinomial distribution θ. The model then generates a multinomial probability conditioned to the chosen topic for the words:

$$z_{ji} = \text{Multinomial}(\theta_j), \theta \sim \text{Dir}(\alpha) \tag{4.3}$$

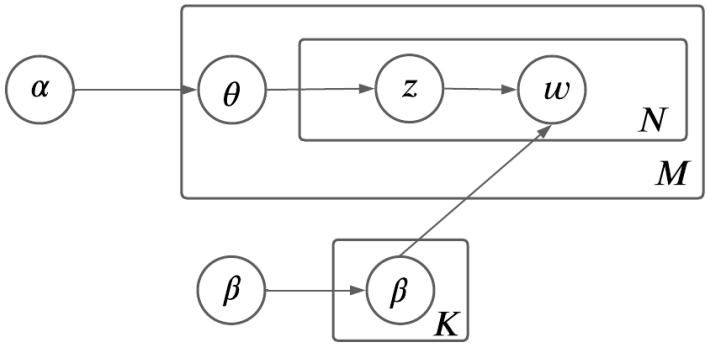
$$w_{ji} \sim p(w_{ji}|z_{ji}, \beta) \tag{4.4}$$

Figure 13 expresses the generative process of LDA: $\text{Dir}(\alpha)$ generates distribution of topics over M documents θ and then distribution of topics over N words ($w \sim \text{Dir}(\beta)$) . In the

end of the process, LDA allows to represent documents as a mixing of topic distributions:

$$p(z,w,\phi,\theta|\alpha,\beta) = \prod_{k=1}^{K} p(\phi_k|\beta) \prod_{j=1}^{M} p(\theta_j|\alpha)[\prod_{i=1}^{V} p(z_{ji}|\vec{\theta}) p(w_{ij}|z_{ij},\phi_{z_{ji}})] \tag{4.5}$$

Figure 13 – Plate notation for LDA



Source: Adapted from Blei, Ng and Jordan (2003).

Nonnegative Matrix Factorization (NMF) (LOPES; RIBEIRO, 2015) is a method that consists in factorizing a matrix with non-negative elements into new matrices, also with non-negative values. NMF was shown to be analytically equivalent LDA (FALEIROS; LOPES, 2016). When applied to Topic Modelling, NMF factorizes a document-term matrix $\mathbf{F}_{D\times T}$ into matrices $\mathbf{W}_{D\times K}$ and $\mathbf{H}_{K\times T}$ by minimizing a loss function expressed as distance metric:

$$\text{loss} = f(|F - WH|^2) \tag{4.6}$$

for which the discrete KL-Divergence is well suited:

$$\text{loss} = Q_{KL-NMF} = \sum_{ji}(F_{ji}\log\frac{F_{ji}}{[\mathbf{WH}]_{j,i}} - F_{ji}[\mathbf{WH}]_{ji}) \tag{4.7}$$

The optimal solution for equation 4.7 is found by using gradient descent to compute the update update rules for elements of factor matrices

$$W_{jk} = W_{jk}\frac{\sum_{j,i} W_{ki}F_{ji}/[\mathbf{WH}]_{ji}}{\sum_q H_{kq}} \tag{4.8}$$

$$H_{jk} = H_{jk}\frac{\sum_{ji} W_{ki}F_{ji}/[\mathbf{WH}]_{ji}}{\sum_q W_{kq}} \tag{4.9}$$

A common critique about the models presented above is the fact that words are considered to be interchangeble with the bag-of-words assumption when in fact the linguistic semantics most often relies on word dependencies. Specific deep learning models regarding texts as sequences

and modelled in layers of recurrent (DIENG *et al.*, 2016), autoregressive (LAROCHELLE; LAULY, 2012), (GUPTA *et al.*, 2019) and other topologies. Such models aim to use the structural dependencies of words in a sentence for better performance. Pure neural network approaches to Topic Modelling are out of the scope of this work.

4.2 Pre-trained Models

Vector representation for text data works using the Encoder-Decoder mechanism *Bidirectional Encoder Representation Transformers* (BERT) (DEVLIN *et al.*, 2018) is a model that learns representation for unlabeled text using 30,000 token vocabulary embedding and can be pre-trained to generate predictions for different types of tasks (*e.g.* translation, topic modelling) and later fine-tuned with labeled data.

GPT (Generative Pre-trained Transformers) (RADFORD *et al.*, 2018) is another example of pre-trained transformer-based model generator for text processing and topic modelling. It takes an unsupervised corpus as input and passes it through a neural network in order to maximize the probability likelihood for the decoder. Its current installment, GPT-4 (OPENAI, 2023), is pre-trained with a large amount of text data and is regarded as a very precise text generation model.

Both pre-trained models described have achieved state-of-the-art performance for text-related tasks. An in-depth analysis of these models is out of the scope of this dissertation.

4.3 Graph Representation for Topic Modelling

The usage of graph representation for classification tasks (such as topic detection) has already been explored mainly in semi-supervised learning with the transductive set. This section brings an example of heuristics-based techniques.

Propagation on Bi-partite Graphs (PBG) (FALEIROS; VALEJO; LOPES, 2020) is a unsusupervised algorithm that propagates latent variables of the words and documents in neighbourhoods. The collection of documents is represented as a bi-partite graph $G(\mathscr{D} \cup \mathscr{W}, \mathscr{E}, f)$ where the vertices are the documents (set $\mathscr{D}$) and the words (set $\mathscr{W}$) linked by an edge $e_{ji} \in \mathscr{E}$. Frequency of word w_i in document $\mathbf{w}_j$ is represented by f_{ji}. Given the number of topics K_z PBG propagates multidimensional latent variables $A_j, B_i, C_{e_{ji}}$ associated with the vertices and the edges that at first are initialized randomly constrained to $\sum_{k=1}^{K} A_{jk} = 1$ and $\sum_{w_i \in \mathscr{W}} B_{ik} = 1$. These latent variables are then updated by a local propagation for each edge

$$C_{e_{ji}} = A_j \odot B_i \tag{4.10}$$

constrained to $\sum_{k=1}^{K_z} C_{e_{ji}} = 1$. The values computed in the local propagation are fed again to A_j, including the α parameter to control the concentration of values in A_j.

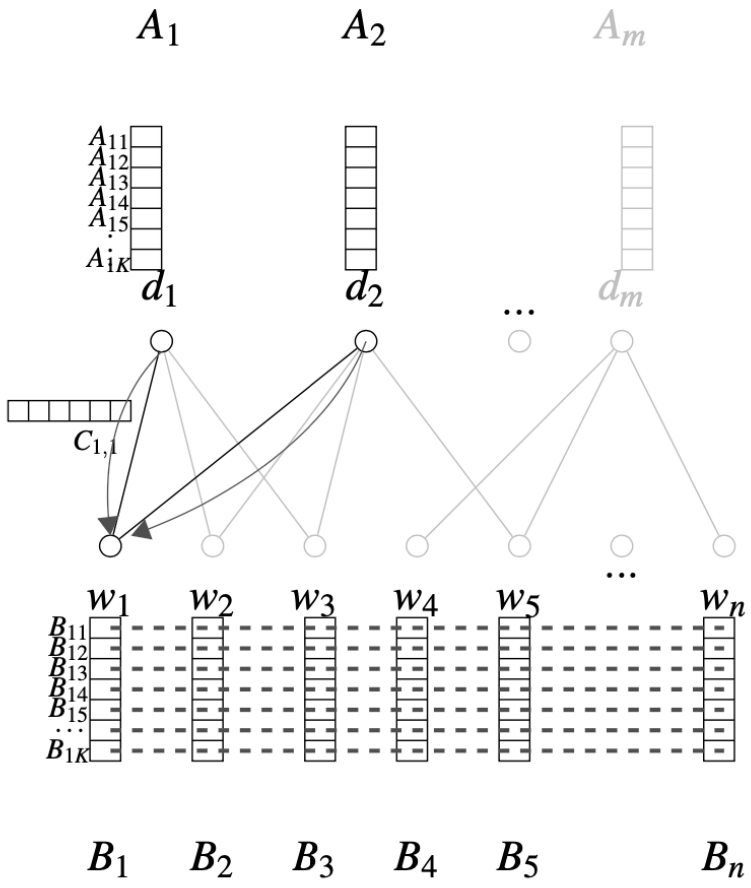
$$A_j = \alpha + \sum_i f_{ji} C_{e_{ji}} \tag{4.11}$$

A global propagation step takes place for each word vertex and its edges.

$$B_i = \sum_{\mathbf{w}_j \in \mathscr{D}} f_{ji} C_{e_{ji}} \tag{4.12}$$

At the end of the process, PBG models the influence of each word w_i from a document $\mathbf{w}_j$ in the array A_j. For topics, the influence of w_i is given by the array B_i.

Figure 14 – PBG representation of text



Source: Adapted from Faleiros, Valejo and Lopes (2020).

KL-Divergence between vectors is used to find the optimal.

$$\text{loss} = Q_{KL-G(A,B,C)} = \sum_{e_{ji} \in \mathscr{E}} [f_{ji} C_{e_{ji}} \log \frac{A_j \odot B_i}{C_{e_{ji}}} + \sum_{\mathbf{w}_{j \in \mathscr{D}}} \mathscr{R}(A_j, \alpha)] \tag{4.13}$$

where the regularization factor, applied to each document, $\mathscr{R}$ is given by

$$\mathscr{R}(A_j, \alpha) = (\alpha - A_{jk}) \log A_{jk} + A_{jk} \log(A_{jk} - 1), k = 1, ...K \tag{4.14}$$

Transductive Propagation on Bi-partite Graphs (TPBG) (FALEIROS; VALEJO; LOPES, 2020) is a semi-supervised variant of PBG. It applies the transductive learning approach and makes an estimation of labels for unlabeled documents $\mathscr{D}^u$ using labeled documents $\mathscr{D}^l$ without creating predictive model. Documents and words compose a bi-partite graph as in PBG. For TPBG the discriptive vectors have the same dimension size as the number of classes from the dataset. Equation 4.13 is changed to

$$\text{loss} = Q_{KL-G(A,B,C)} = \sum_{e_{ji} \in \mathscr{E}} [f_{ji} C_{e_{ji}} \log \frac{A_j \odot B_i}{C_{e_{ji}}} + \sum_{\mathbf{w}_{j \in \mathscr{D}}} \mathscr{R}(A_j, \alpha) + \sum_{\mathbf{w}_{j \in \mathscr{D}^l}} y_j \log \frac{A_j}{y_j}] \tag{4.15}$$

where the regularization term $\mathscr{R}(A_j, \alpha)$ is given by

$$\mathscr{R}(A_j, \alpha) = (\alpha - A_j) \log A_j + A_j \log(A_j - 1) \tag{4.16}$$

and the term $\sum_{\mathbf{w}_{j \in \mathscr{D}^l}} Y_j \log \frac{A_j}{Y_j}$ is set to ensure the likelihood of the estimated label in respect to the labeled documents. TPBG achieved the state-of-the-art performance on semi-supervised text classification tasks.

4.4 GNN for Topic Modelling

The growing number of publications on Graph Neural Networks in recent years had also contributed to Topic Modelling research. While the aforementioned models were based on latent variables computed with the bag-of-words assumption over the documents, models based on GNNs leverage the neighbourhood structure information from the documents (and also words) represented as nodes in a graph. Relevant GNN-based topic models from the literature are discussed in this section.

Graph Topic Model (GTM) (ZHOU; HU; WANG, 2020) proposes to represent the corpus $\mathscr{D}$ as an undirected graph with words and documents as nodes. There are $N + M$ nodes in total

(sum of the number of documents in $\mathscr{D}$ and the size of the vocabulary $\mathscr{V}$). Edges are created based on co-occurrences of words with TF-IDF metric:

$$A_{ij} = \begin{cases} \text{TF-IDF}_{ij} \text{ if } i \in D \text{ and } j \in \mathscr{V} \\ \text{TF-IDF}_{ji} \text{ if } i \in \mathscr{V} \text{ and } j \in D \\ 1 \text{ if } i = j \text{ (auto-connection)} \\ 0 \text{ otherwise} \end{cases} \tag{4.17}$$

Node features are given by the latent topic variables. The topical representation of a document is then aggregated by message passing including word and document nodes and encoded into $\hat{\mathbf{Z}}$ using GCN framework given by equation 3.13. Later, the word weights $\hat{\mathbf{A}}$ are decoded using an MLP.

$$\hat{\mathbf{Z}} = \text{CGN}(\mathbf{A}) \text{ (Encoder)} \tag{4.18}$$

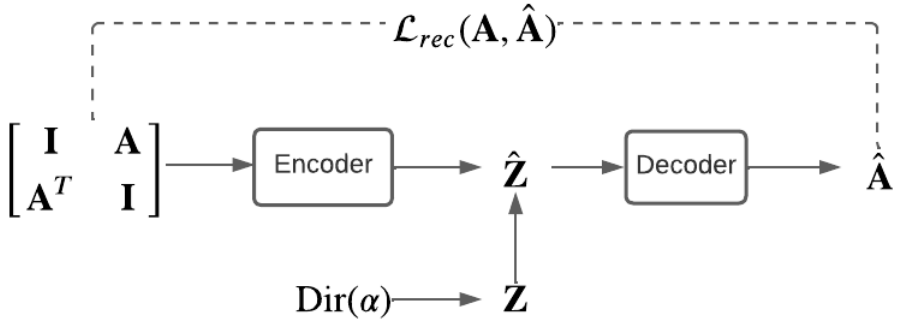
$$\hat{\mathbf{A}} = \text{MPL}(\hat{\mathbf{Z}}, \text{Dir}(\alpha)) \text{ (Decoder)} \tag{4.19}$$

The training objective is learned by optimizing the *document reconstruction loss function*:

$$\text{loss}_{rec}(\mathbf{A}, \hat{\mathbf{A}}) = -\text{mean}[x \log(\hat{x})] \tag{4.20}$$

where x and $\hat{x}$ are the TF-IDF of a document and its reconstructed word distribution. GTM can be expressed as an Encode-Decoder schema as seen in Figure 15, where

Figure 15 – Graph Topic Model diagram



Source: Adapted from Zhou, Hu and Wang (2020).

Graph Attention Topic Modeling Networks (GATON) (YANG *et al.*, 2020) analyses similarities between pLSI (referred to as Stochastic Block Models) on bi-partite graphs and GAT to propose a model that generates a bi-partite graph with documents and words as node

types. Word node attributes are given by embeddings functions and document node attributes are learned by the bi-partite graph edges, represented as a frequency vector, where the frequency of word w_i in document $\mathbf{w}_j$ is represented by f_{ji}:

$$x_{w_i} = \text{embedding}(w_i) \tag{4.21}$$

$$x_{\mathbf{w}_j} = (f_{j1}, ..., f_{jN}) \tag{4.22}$$

Different embedding dimensions are possible for document and word representations as nodes of the bi-partite graph, leading to different unsymmetrical attention normalization constants $\alpha_{doc-word}$ and $\alpha_{word-doc}$. GATON has the basic nodes representation as

$$h_{word} = \varphi(\sum_{t \in \mathcal{N}_{word}} \alpha_{word-doc} h_t), h_{doc} = \varphi(\sum_{z \in \mathcal{N}_{doc}} \alpha_{doc-word} h_z) \tag{4.23}$$

Reconstruction loss function $\mathscr{L}_{rec}$ is then used on the final GNN layers with the final representation of words and documents to achieve the optimal solution.

Graph Neural Topic Model (GNTM) (SHEN *et al.*, 2021) uses multinomial distributions on vocabulary and word sets to build topics and generate a graph. Neural Variational Inference is then used to learn a GNN as an encoder-decoder set minimizing ELBO loss function. GNTM is presented as a special case of LDA.

MagNet (ZHANG *et al.*, 2021) uses the mathematical properties of the hermitian matrix (called magnetic Laplacian) to produce a GAT network that learns both attention weights and the adjacency matrix. It uses BERT word embeddings to train the model.

4.5 Summary

Topic Modelling is a very common text-processing problem in Computer Science. In this section the main concepts of the Topic Modelling problem were covered, from the initial models based on the bag-of-words assumption to probabilistic models, graph-based algorithms and finally GNN-based models.

The next chapter presents this dissertation's proposed approach to text classification using topic modelling embeddings for Graph Neural Networks.

CHAPTER

5

PROPOSAL AND RESULTS

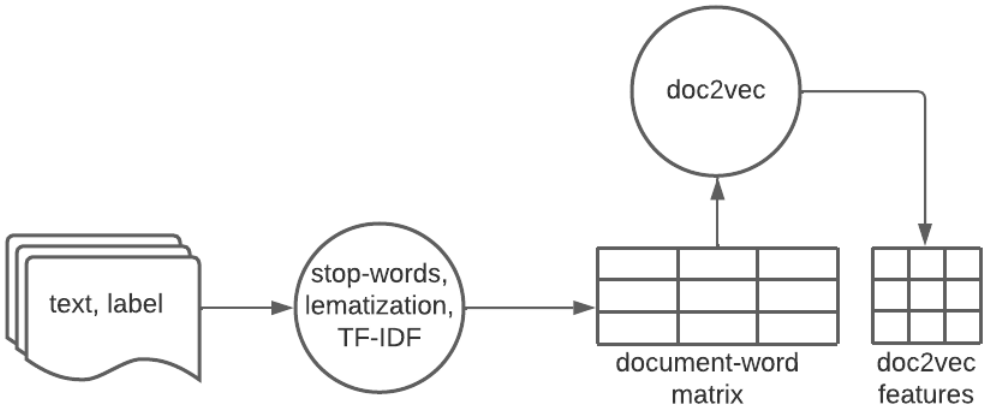
5.1 Proposed Model

The proposed experiment is designed to test a hybrid model that leverages bi-partite graph representation of the corpus computed by TPBG algorithm (FALEIROS; VALEJO; LOPES, 2020) and the GraphSAGE (HAMILTON; YING; LESKOVEC, 2017), adding document descriptive features from doc2vec (LE; MIKOLOV, 2014). This model learns text representations in a two-step training process and predicts the class of documents. The main reason for modelling the problem with these algorithms is to use the bi-partite document-word graph structure generated from text without having convert it into a homogeneous graph.

TPBG is an semi-supervised algorithm that estimated topics for unlabeled documents via propagation of information on the neighbourhood of a bi-partite graph. It was developed by a member of the research group of this master's degree candidate. GraphSAGE was chosen as a GNN model for its capability of dealing with large graphs, such as text-based graphs. Doc2vec is an algorithm that learns paragraph and document vector representations with memory-efficient bag-of-words model.

The first stage of the training process if training TPBG. The text collection has documents marked as labeled $\mathscr{D}^l$ (training set) and unlabeled $\mathscr{D}^u$ (validation set). The corpus $\mathscr{D} = \mathscr{D}^u \cup \mathscr{D}^l$ is pre-processed to be removed from common, low-value words (known as stop-words) lower-case standardized and lemmatized (different forms of a word are reduced to a single one). The pre-processed text is later represented in a vector space with IF-IDF. Doc2vec is used to process the dataset and produce a set of 400 descriptive features. This process is shown in Figure 16.

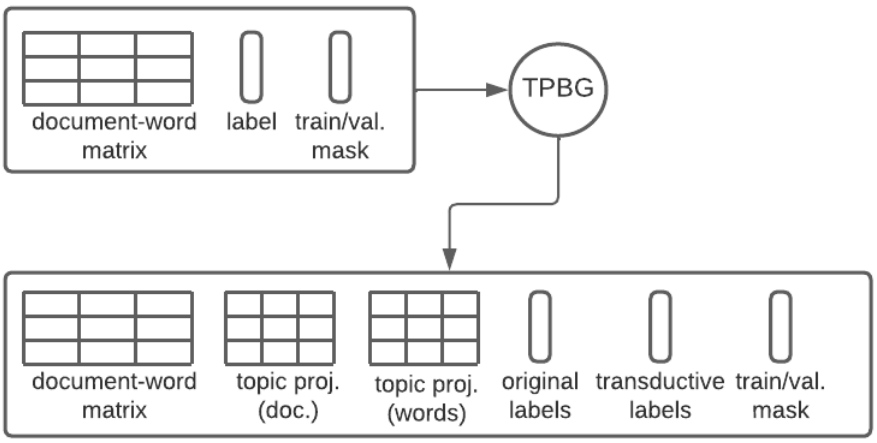
Figure 16 – Text processing for the proposed model



Source: Elaborated by the author.

The process of fitting TPBG to corpus is as follows: each document is represented as a row in a TF-IDF matrix generated by the pre-prossessing phase. The documents from the dataset are marked with a binary mask that indicates the training set and the validation set. For the training of TPBG, the documents from the validation set are used as the unlabeled documents. The class value of the unlabeled documents is initialized with a proxy value of -1. The algorithm propagates the descriptive vectors for documents and words via the edges of the bi-partite graph. By the end of the process all documents have descriptive vectors optimized by TPBG. The A and B vectors (equations 4.11 and 4.12) are stored for each document, representing the document and word features of the graph. This process is shown in Figure 17.

Figure 17 – TPBG training



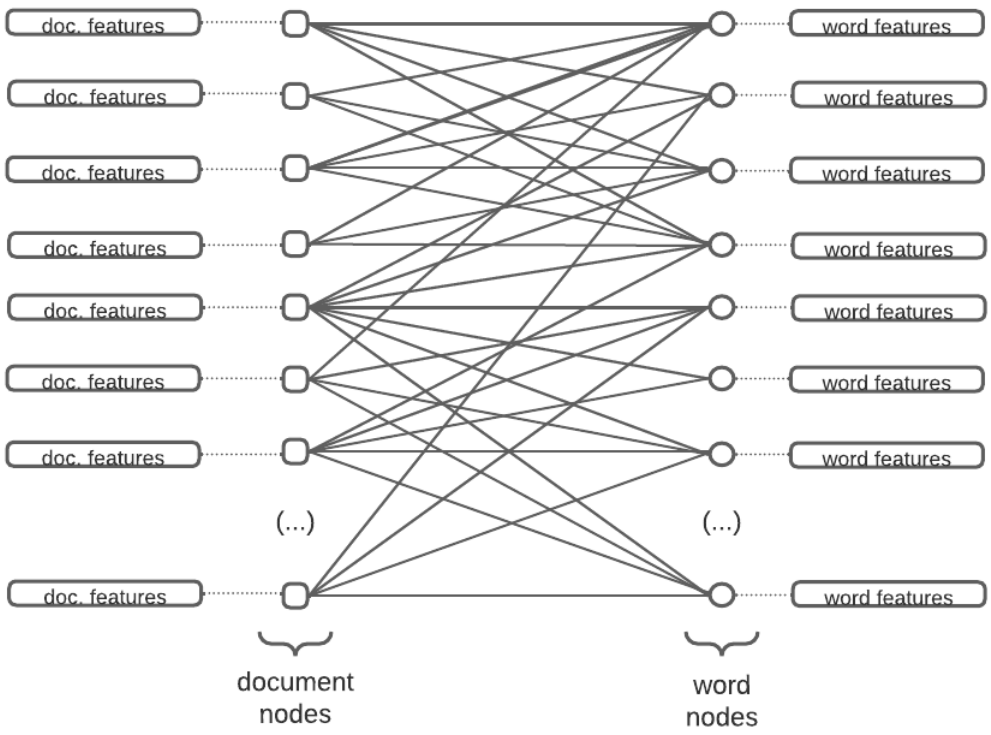
Source: Elaborated by the author.

For each document, doc2vec algorithm is executed and the result features are concatenated with the z topic projections for documents calculated by TPBG generating $\mathbf{X_w}$ the document feature vectors. The z topic projections for words given by TPBG, represented by a $\mathbf{X}_w$ is used by the proposed model as word features.

Based on the mask applied to identify the labeled and unlabeled documents, the corpus is divided into training and validation sets. Both sets carry the original labels to the next stage. Labels calculated by TPBG for the unlabeled documents are discarted.

One bi-partite graph is then generated from each dataset using the TPBG document-word matrix structure to create document-word edges. Document and word features are associated with the nodes of the bi-partite graphs as shown in Figure 18.

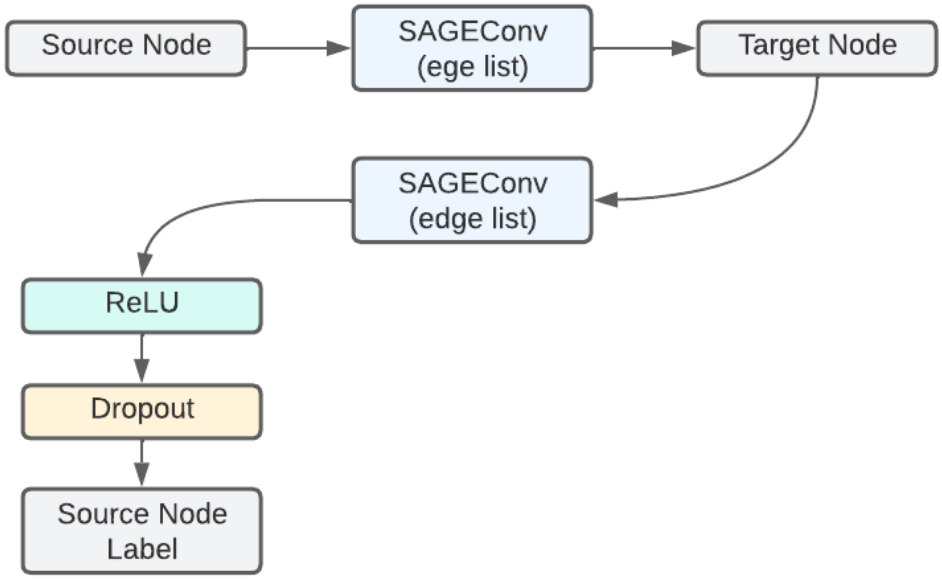
Figure 18 – Bi-partite graph representation of text



Source: Elaborated by the author.

The second training stage is when GraphSAGE is trained on the bi-partite graph from the training dataset. As there is a predictive GNN model generated, this phase is an example of inductive learning. The neural network is allowed to have an arbitrary number of hidden channels and hidden layers. The output layer has the dimension of the number of labels observed during training. The training strategy for one GNN layer is shown in Figure 19.

Figure 19 – Bi-partite GNN learning



Source: Adapted from Fey and Lenssen (2019).

where document nodes (denoted by source nodes) are associated with the topic values. Documents are linked to word nodes (target by source) by the edge list. GraphSAGE convolutions pass the message of node summarization via the edge list of the two node types over K iterations to get the final node embeddings. A dropout regularization layer is added to avoid overfitting GNN learning. The objective of the training process the GNN to learn the document class representation from the training graph, and then evaluate the predictions of the model on the validation graph.

5.2 Experimental Setup

The experiments were performed on 11 opensource benchmark datasets: 20 News Groups (RENNIE, 2008), BBC News (GREENE; CUNNINGHAM, 2006), Classic4 (M., 2010), Computer Science Technical Reports Collection (CSTR) (JONES; CUNNINGHAM; MCNAB, 1998), National Science Foundation Collection (NSF) (PAZZANI, 2017), Web Knowledge Base (WebKB) Collection (CRAVEN *et al.*, 1998), Re8 Collection (FORMAN, 2006) and four datasets from the Directory Mozilla The Open Directory Project (DMOZ) (GRAHAM, 2004): DMOZ Computers, DMOZ Health, DMOZ Science and DMOZ Sports. Preprocessing text data used word lemmatizer from NLTK (XUE, 2011) and IT-IDF vectorizer from ScikitLearn (BUITINCK *et al.*, 2013). Table 3 presents for each dataset: M (number of documents), N (number of terms), $\bar{N}$ (average number of terms per document) and C (number of classes).

Table 3 – List of benchmark datasets

Dataset	M	N	$\bar{N}$	C
20 News Groups	18808	45434	76.47	20
BBC News	2225	9635	197.07	5
Classic4	7095	7749	35.28	4
CSTR	299	5427	57.27	4
NSF	10524	3888	6.56	16
WebKB	8282	22892	89.78	7
Re8	7674	3531	35.31	8
DMOZ Computers	95000	5111	10.83	19
DMOZ Health	6500	4217	12.40	13
DMOZ Science	6000	4821	11.52	12
DMOZ Sports	13500	5682	11.87	27

Source: Adapted from Rossi, Marcacini and Rezende (2013).

In the experimental scenarios the hybrid model overfitted when the corpus had a number of classes greater than 5. In order to perform similar classification tasks across all datasets, five classes were selected from the datasets to perform the training for all experiments.

A 60% fraction of each dataset was marked as labeled. The TPBG algorithm was then executed on the dataset with α parameter set to 0.5%, 50 local iterations and 10 global iterations. The Gensim Project (REHUREK; SOJKA, 2011) implementation of Doc2vec is processed in the corpus to produce document features with the same structure both for train and test datasets.

To generate the bi-partite graph representation the Heterodata class from Pytorch Geometric (FEY; LENSSEN, 2019) was used as a wrapper. All convolutions were initialized with lazy evaluations (without specifying input dimensionality).

The neural network input layer reads input signals from the bi-partite nodes. Document features $\mathbf{X_w}$ and word features are $\mathbf{X}_w$ feed the GNN layers (stacked with HeteroConv method by Pytorch Geometric) calculating node embeddings using *sum* as aggregate function. The output layer is composed of *softmax* projection with dimension equal to the number of classes to be modelled. The activation function used for the convolution layers is LeakyReLU, a special case of the parametric ReLU from Table 1 where α is a hyperparameter learned by the network instead of a given value. The stochastic optimization engine used was based ADAM method (KINGMA; BA, 2014).

Two functions were tested as loss functions: Cross Entropy (CE) and Focal Loss (FL):

$$loss_{CE} = -\sum_{i=1}^{i=C} y_i log(\text{softmax}(i)) \tag{5.1}$$

$$loss_{FL} = -\sum_{i=1}^{i=C} y_i (1-\text{softmax}(i))^{\gamma} log(\text{softmax}(i)) \tag{5.2}$$

where C is the number of classes for the dataset and γ is is a hyperparameter with value set to 1 in the experiments.

The training process selected the best model from the execution of 1,500 epochs and is considered to achieve convergence for the minimum value of the loss function after all the epochs. Weight decay and dropout were used as regularization techniques for the GNN.

The experiment setup is a multi-class classification with a semi-supervised training stage (TPBG) and a supervised training stage (GNN). To evaluate performance on class predictions on test dataset the metric chosen was micro-averaging F1-score (F_1^{μ}):

$$F_1^{\mu} = \frac{2P^{\mu}R^{\mu}}{P^{\mu} + R^{\mu}} \tag{5.3}$$

where P^{μ} is the micro-averaging precision (average, for each class, of true positive count divided by the count of positive and negative examples) and R^{μ} is the micro-averaging recall (average, for each class, of true positive count divided by the count positive examples). F_1^{μ} is commonly used in multi-class classification to measure harmonic average of precision and recall.

All experiments were performed using a 2,7GHz Quad-Core Intel Core i7 processor with 16GB of memory. The source code of the experiments is available on the project page on GitHub (<https://github.com/brunofbessa/masters_degree_icmc/tree/master/project>).

The experimental parameters are listed below:

Table 4 – List of experimental parameters

Parameter	Context	Vaues
$\lvert\mathbf{x}_W\rvert$	Number word features from TPBG	z
$\lvert\mathbf{x}_D\rvert$	Number of document features form TPBG and Doc2vec	400 + z
K	Number of convolutional layers	2, 3, 4
n_h	Number of channels on hidden layers	20, 40, 100
λ_{L2}	L2 regularization factor	0.00001
η	GNN training learning rate	0.001
$p_{dropout}$	Dropout probability	0, 20%
$loss$	Loss function	CE, FL

Source: Research data.

5.3 Experimental Results

All the different experimental setups from Table 4 are presented for each dataset in Tables 5 to 15. Table 16 has a resume of the best performance from these scenarios for each dataset.

Table 5 – Experimental results for dataset 20 News Groups

n_h	K	Dropout Prob. (%)	Loss Function	F_1^{μ} (%)	Num. Epochs
20	2	0	CE	79.69	1412
20	2	0	FL	79.41	739
20	2	0.2	CE	79.77	869
20	2	0.2	FL	79.60	737
20	3	0	CE	79.73	643
20	3	0	FL	79.37	1016
20	3	0.2	CE	79.39	1286
20	3	0.2	FL	79.54	1239
20	4	0	CE	79.55	1210
20	4	0	FL	79.6	870
20	4	0.2	CE	79.53	993
20	4	0.2	FL	79.14	1379
40	2	0	CE	79.54	296
40	2	0	FL	79.53	405
40	2	0.2	CE	79.69	1087
40	2	0.2	FL	79.56	1218
40	3	0	CE	79.45	155
40	3	0	FL	79.67	349
40	3	0.2	CE	79.98	873
40	3	0.2	FL	79.62	330
40	4	0	CE	79.94	412
40	4	0	FL	79.78	353
40	4	0.2	CE	79.93	1308
40	4	0.2	FL	79.77	1488
100	2	0	CE	79.72	775
100	2	0	FL	79.33	752
100	2	0.2	CE	80.10	859
100	2	0.2	FL	79.85	291
100	3	0	CE	80.19	393
100	3	0	FL	79.80	309
100	3	0.2	CE	80.18	986
100	3	0.2	FL	80.11	893
100	4	0	CE	80.17	387
100	4	0	FL	80.06	426
100	4	0.2	CE	**80.37**	877
100	4	0.2	FL	80.19	467

Source: Elaborated by the author.

Table 6 – Experimental results for dataset BBC News

n_h	K	Dropout Prob. (%)	Loss Function	F_1^{μ} (%)	Num. Epochs
20	2	0	CE	97.53	1446
20	2	0	FL	97.19	1499
20	2	0.2	CE	97.86	1241
20	2	0.2	FL	97.64	1367
20	3	0	CE	96.86	801
20	3	0	FL	97.53	1499
20	3	0.2	CE	97.64	1292
20	3	0.2	FL	97.75	1437
20	4	0	CE	97.63	708
20	4	0	FL	97.52	1183
20	4	0.2	CE	97.30	1428
20	4	0.2	FL	97.07	1443
40	2	0	CE	97.53	1499
40	2	0	FL	97.53	1070
40	2	0.2	CE	97.75	1447
40	2	0.2	FL	97.98	1403
40	3	0	CE	97.63	1218
40	3	0	FL	97.30	223
40	3	0.2	CE	**97.97**	1064
40	3	0.2	FL	97.86	1251
40	4	0	CE	97.31	632
40	4	0	FL	97.52	891
40	4	0.2	CE	97.97	1326
40	4	0.2	FL	97.64	1399
100	2	0	CE	97.52	964
100	2	0	FL	97.53	637
100	2	0.2	CE	97.97	1309
100	2	0.2	FL	97.98	1174
100	3	0	CE	97.63	604
100	3	0	FL	97.52	737
100	3	0.2	CE	97.97	558
100	3	0.2	FL	97.74	1222
100	4	0	CE	97.64	191
100	4	0	FL	97.63	835
100	4	0.2	CE	97.75	903
100	4	0.2	FL	97.86	1002

Source: Elaborated by the author.

Table 7 – Experimental results for dataset Classic4

n_h	K	Dropout Prob. (%)	Loss Function	F_1^μ (%)	Num. Epochs
20	2	0	CE	95.49	1151
20	2	0	FL	95.49	1499
20	2	0.2	CE	95.31	1439
20	2	0.2	FL	94.85	1471
20	3	0	CE	95.87	934
20	3	0	FL	95.56	777
20	3	0.2	CE	95.27	1469
20	3	0.2	FL	95.09	1417
20	4	0	CE	95.73	608
20	4	0	FL	95.63	1044
20	4	0.2	CE	95.30	1175
20	4	0.2	FL	95.25	1342
40	2	0	CE	95.61	1095
40	2	0	FL	95.71	1064
40	2	0.2	CE	95.45	1321
40	2	0.2	FL	95.38	884
40	3	0	CE	95.70	628
40	3	0	FL	95.63	630
40	3	0.2	CE	95.49	1106
40	3	0.2	FL	95.51	1078
40	4	0	CE	95.62	542
40	4	0	FL	95.52	395
40	4	0.2	CE	95.37	929
40	4	0.2	FL	95.57	912
100	2	0	CE	95.77	748
100	2	0	FL	95.74	650
100	2	0.2	CE	95.62	740
100	2	0.2	FL	95.52	668
100	3	0	CE	95.87	470
100	3	0	FL	95.70	419
100	3	0.2	CE	95.66	445
100	3	0.2	FL	95.76	421
100	4	0	CE	**95.87**	502
100	4	0	FL	95.62	270
100	4	0.2	CE	95.72	538
100	4	0.2	FL	95.83	343

Source: Elaborated by the author.

Table 8 – Experimental results for dataset NSF

n_h	K	Dropout Prob. (%)	Loss Function	F_1^{μ} (%)	Num. Epochs
20	2	0	CE	89.26	267
20	2	0	FL	89.48	111
20	2	0.2	CE	89.33	1196
20	2	0.2	FL	89.56	1193
20	3	0	CE	89.63	122
20	3	0	FL	89.56	808
20	3	0.2	CE	89.26	1469
20	3	0.2	FL	89.93	765
20	4	0	CE	89.71	269
20	4	0	FL	89.33	528
20	4	0.2	CE	89.63	1344
20	4	0.2	FL	89.78	1169
40	2	0	CE	89.48	84
40	2	0	FL	89.78	1147
40	2	0.2	CE	89.93	1474
40	2	0.2	FL	90.01	795
40	3	0	CE	89.56	341
40	3	0	FL	89.56	755
40	3	0.2	CE	89.93	1386
40	3	0.2	FL	90.08	1301
40	4	0	CE	89.86	279
40	4	0	FL	89.63	429
40	4	0.2	CE	89.93	1310
40	4	0.2	FL	89.86	503
100	2	0	CE	89.78	811
100	2	0	FL	89.93	766
100	2	0.2	CE	89.93	1360
100	2	0.2	FL	90.01	802
100	3	0	CE	89.78	271
100	3	0	FL	89.78	279
100	3	0.2	CE	90.08	937
100	3	0.2	FL	**90.16**	386
100	4	0	CE	89.86	32
100	4	0	FL	89.48	152
100	4	0.2	CE	90.01	403
100	4	0.2	FL	90.16	647

Source: Elaborated by the author.

Table 9 – Experimental results for dataset WebKB

n_h	K	Dropout Prob. (%)	Loss Function	F_1^μ (%)	Num. Epochs
20	2	0	CE	75.57	1498
20	2	0	FL	75.56	1267
20	2	0.2	CE	71.59	1458
20	2	0.2	FL	70.48	1479
20	3	0	CE	77.30	1141
20	3	0	FL	78.97	1221
20	3	0.2	CE	74.33	1496
20	3	0.2	FL	72.72	1478
20	4	0	CE	79.09	1153
20	4	0	FL	77.85	1497
20	4	0.2	CE	75.03	1429
20	4	0.2	FL	73.26	1388
40	2	0	CE	76.69	1102
40	2	0	FL	77.07	1157
40	2	0.2	CE	75.00	1480
40	2	0.2	FL	74.98	1462
40	3	0	CE	79.23	872
40	3	0	FL	78.16	1066
40	3	0.2	CE	76.94	1100
40	3	0.2	FL	76.57	1298
40	4	0	CE	80.16	814
40	4	0	FL	79.68	1054
40	4	0.2	CE	77.94	1397
40	4	0.2	FL	76.89	1469
100	2	0	CE	78.04	783
100	2	0	FL	77.72	628
100	2	0.2	CE	76.67	1078
100	2	0.2	FL	76.48	1109
100	3	0	CE	78.82	800
100	3	0	FL	80.16	629
100	3	0.2	CE	78.67	658
100	3	0.2	FL	78.78	813
100	4	0	CE	79.92	679
100	4	0	FL	80.75	731
100	4	0.2	CE	**81.10**	827
100	4	0.2	FL	80.90	1187

Source: Elaborated by the author.

Table 10 – Experimental results for dataset CSTR

n_h	K	Dropout Prob. (%)	Loss Function	F_1^{μ} (%)	Num. Epochs
20	2	0	CE	76.71	222
20	2	0	FL	75.92	203
20	2	0.2	CE	75.70	721
20	2	0.2	FL	73.99	326
20	3	0	CE	74.26	198
20	3	0	FL	71.1	82
20	3	0.2	CE	74.65	166
20	3	0.2	FL	73.27	179
20	4	0	CE	68.51	103
20	4	0	FL	72.55	308
20	4	0.2	CE	73.02	156
20	4	0.2	FL	70.35	115
40	2	0	CE	75.96	121
40	2	0	FL	75.52	107
40	2	0.2	CE	74.69	1057
40	2	0.2	FL	77.62	126
40	3	0	CE	77.45	81
40	3	0	FL	**79.54**	50
40	3	0.2	CE	71.47	84
40	3	0.2	FL	73.20	82
40	4	0	CE	68.80	560
40	4	0	FL	69.51	71
40	4	0.2	CE	73.26	83
40	4	0.2	FL	72.48	62
100	2	0	CE	74.69	53
100	2	0	FL	76.52	53
100	2	0.2	CE	77.62	59
100	2	0.2	FL	75.37	57
100	3	0	CE	73.01	28
100	3	0	FL	74.40	44
100	3	0.2	CE	73.36	40
100	3	0.2	FL	77.69	41
100	4	0	CE	69.57	29
100	4	0	FL	73.39	36
100	4	0.2	CE	72.54	29
100	4	0.2	FL	73.20	31

Source: Elaborated by the author.

Table 11 – Experimental results for dataset Re8

n_h	K	Dropout Prob. (%)	Loss Function	F_1^μ (%)	Num. Epochs
20	2	0	CE	93.58	1498
20	2	0	FL	93.39	1485
20	2	0.2	CE	91.87	1496
20	2	0.2	FL	91.62	1470
20	3	0	CE	93.90	1082
20	3	0	FL	94.08	1283
20	3	0.2	CE	91.38	1458
20	3	0.2	FL	91.29	1494
20	4	0	CE	93.34	863
20	4	0	FL	93.47	1044
20	4	0.2	CE	89.82	1492
20	4	0.2	FL	91.24	1426
40	2	0	CE	94.40	1486
40	2	0	FL	94.39	1401
40	2	0.2	CE	93.13	1493
40	2	0.2	FL	93.43	1454
40	3	0	CE	94.39	797
40	3	0	FL	94.18	859
40	3	0.2	CE	93.83	1138
40	3	0.2	FL	93.86	1170
40	4	0	CE	94.13	736
40	4	0	FL	94.31	696
40	4	0.2	CE	93.72	1381
40	4	0.2	FL	93.53	1164
100	2	0	CE	94.54	869
100	2	0	FL	94.61	934
100	2	0.2	CE	94.52	1122
100	2	0.2	FL	94.38	1224
100	3	0	CE	94.67	430
100	3	0	FL	94.66	613
100	3	0.2	CE	94.56	762
100	3	0.2	FL	94.43	896
100	4	0	CE	94.52	508
100	4	0	FL	**94.69**	517
100	4	0.2	CE	94.47	660
100	4	0.2	FL	94.44	864

Source: Elaborated by the author.

Table 12 – Experimental results for dataset DMOZ Computers

n_h	K	Dropout Prob. (%)	Loss Function	F_1^μ (%)	Num. Epochs
20	2	0	CE	80.62	370
20	2	0	FL	80.73	187
20	2	0.2	CE	81.33	537
20	2	0.2	FL	81.55	1282
20	3	0	CE	80.46	307
20	3	0	FL	80.88	989
20	3	0.2	CE	81.50	1015
20	3	0.2	FL	81.63	1050
20	4	0	CE	80.60	173
20	4	0	FL	80.85	690
20	4	0.2	CE	81.41	1399
20	4	0.2	FL	81.55	1327
40	2	0	CE	80.69	80
40	2	0	FL	81.32	132
40	2	0.2	CE	81.27	1006
40	2	0.2	FL	81.49	436
40	3	0	CE	80.58	160
40	3	0	FL	80.84	116
40	3	0.2	CE	81.60	1217
40	3	0.2	FL	81.56	305
40	4	0	CE	80.62	93
40	4	0	FL	80.73	394
40	4	0.2	CE	81.61	1072
40	4	0.2	FL	**81.88**	905
100	2	0	CE	80.66	1448
100	2	0	FL	80.97	727
100	2	0.2	CE	81.76	1237
100	2	0.2	FL	81.44	1276
100	3	0	CE	80.62	226
100	3	0	FL	80.92	253
100	3	0.2	CE	81.39	454
100	3	0.2	FL	81.46	491
100	4	0	CE	80.89	42
100	4	0	FL	80.69	182
100	4	0.2	CE	81.64	220
100	4	0.2	FL	81.68	189

Source: Elaborated by the author.

Table 13 – Experimental results for dataset DMOZ Health

n_h	K	Dropout Prob. (%)	Loss Function	F_1^μ (%)	Num. Epochs
20	2	0	CE	86.41	525
20	2	0	FL	86.48	1499
20	2	0.2	CE	87.41	1358
20	2	0.2	FL	86.92	1084
20	3	0	CE	87.59	1365
20	3	0	FL	86.89	474
20	3	0.2	CE	87.6	832
20	3	0.2	FL	88.00	1479
20	4	0	CE	87.27	821
20	4	0	FL	87.46	855
20	4	0.2	CE	88.33	1304
20	4	0.2	FL	88.11	1486
40	2	0	CE	86.80	1045
40	2	0	FL	87.27	1314
40	2	0.2	CE	87.51	1363
40	2	0.2	FL	87.52	1453
40	3	0	CE	88.10	779
40	3	0	FL	88.08	773
40	3	0.2	CE	88.69	1472
40	3	0.2	FL	88.66	1408
40	4	0	CE	87.98	911
40	4	0	FL	87.94	650
40	4	0.2	CE	**89.32**	1445
40	4	0.2	FL	89.05	1396
100	2	0	CE	87.87	1010
100	2	0	FL	87.73	1445
100	2	0.2	CE	88.01	1365
100	2	0.2	FL	87.89	1491
100	3	0	CE	88.26	480
100	3	0	FL	88.09	344
100	3	0.2	CE	88.77	1282
100	3	0.2	FL	88.71	664
100	4	0	CE	87.86	330
100	4	0	FL	88.10	835
100	4	0.2	CE	88.90	600
100	4	0.2	FL	89.29	1242

Source: Elaborated by the author.

Table 14 – Experimental results for dataset DMOZ Science

n_h	K	Dropout Prob. (%)	Loss Function	F_1^μ (%)	Num. Epochs
20	2	0	CE	81.73	1499
20	2	0	FL	81.85	1499
20	2	0.2	CE	82.46	721
20	2	0.2	FL	82.25	1298
20	3	0	CE	81.85	1262
20	3	0	FL	81.74	1456
20	3	0.2	CE	**83.16**	1380
20	3	0.2	FL	82.67	1040
20	4	0	CE	82.35	910
20	4	0	FL	82.55	1429
20	4	0.2	CE	82.77	1379
20	4	0.2	FL	82.89	1128
40	2	0	CE	81.63	222
40	2	0	FL	81.85	1160
40	2	0.2	CE	82.55	1261
40	2	0.2	FL	82.56	1310
40	3	0	CE	82.77	869
40	3	0	FL	82.05	731
40	3	0.2	CE	82.96	1048
40	3	0.2	FL	83.03	1118
40	4	0	CE	82.14	390
40	4	0	FL	82.81	542
40	4	0.2	CE	83.16	1471
40	4	0.2	FL	82.97	474
100	2	0	CE	82.14	915
100	2	0	FL	81.86	81
100	2	0.2	CE	83.00	1358
100	2	0.2	FL	82.78	1294
100	3	0	CE	82.81	477
100	3	0	FL	82.05	283
100	3	0.2	CE	82.89	712
100	3	0.2	FL	83.03	906
100	4	0	CE	82.39	434
100	4	0	FL	82.68	843
100	4	0.2	CE	83.00	1246
100	4	0.2	FL	83.07	315

Source: Elaborated by the author.

Table 15 – Experimental results for dataset DMOZ Sports

n_h	K	Dropout Prob. (%)	Loss Function	F_1^μ (%)	Num. Epochs
20	2	0	CE	87.97	1497
20	2	0	FL	88.29	583
20	2	0.2	CE	88.58	439
20	2	0.2	FL	88.51	1281
20	3	0	CE	89.80	1489
20	3	0	FL	89.40	1489
20	3	0.2	CE	88.96	1183
20	3	0.2	FL	89.01	1474
20	4	0	CE	90.05	1118
20	4	0	FL	90.50	1496
20	4	0.2	CE	90.60	1470
20	4	0.2	FL	90.03	1321
40	2	0	CE	88.60	1062
40	2	0	FL	89.90	1376
40	2	0.2	CE	89.39	1483
40	2	0.2	FL	88.90	1357
40	3	0	CE	90.62	1365
40	3	0	FL	91.10	1365
40	3	0.2	CE	91.10	1349
40	3	0.2	FL	90.40	1464
40	4	0	CE	90.09	611
40	4	0	FL	90.91	780
40	4	0.2	CE	91.83	1420
40	4	0.2	FL	92.32	1394
100	2	0	CE	90.21	1334
100	2	0	FL	90.81	1384
100	2	0.2	CE	90.30	1309
100	2	0.2	FL	91.31	1359
100	3	0	CE	91.11	816
100	3	0	FL	92.14	733
100	3	0.2	CE	92.31	1372
100	3	0.2	FL	92.23	1466
100	4	0	CE	90.50	580
100	4	0	FL	92.11	450
100	4	0.2	CE	**92.80**	1475
100	4	0.2	FL	92.40	1399

Source: Elaborated by the author.

In the following table are presented the best performances for each dataset identified as the **Proposed Model**. For comparison effect, table 16 shows the performance of relevant topic modelling techniques from research (LDA and NMF embeddings were used in a random forest to perform classification), as well as BERT, a classification model that reaches the current state-of-the-art.

Table 16 – Classification F1-score (%) Comparison

Dataset	LDA+R. Forest	NMF+R. Forest	TPBG	BERT	Proposed Model
20 News Groups	68.10	74.42	75.64	**98.26**	80.37
BBC News	88.65	93.14	96.42	**99.15**	97.97
Classic4	93.44	94.96	93.84	**98.53**	95.87
CSTR	53.33	79.16	55.61	77.62	**79.54**
NSF	61.68	84.14	86.65	**93.46**	90.16
WebKb	81.14	83.39	50.34	**89.34**	81.10
Re8	92.05	94.52	64.56	**96.49**	94.69
DMOZ Computers	48.73	78.58	79.49	**88.45**	81.88
DMOZ Health	59.70	85.41	84.85	**91.68**	89.32
DMOZ Science	41.15	73.80	80.09	**98.14**	83.16
DMOZ Sports	76.26	93.52	85.07	**98.05**	92.80

Source: Elaborated by the author.

LDA and NMF models were fit to the datasets using scikit-learn implementations. The parameters used for both models were the same (the number of topics was set to 100). The embedding from LDA and NMF were used as features for a Random Forest classifier (also from scikit-learn implementation), with the number of estimators set to 1000. The same TPBG setup described in section 5.2 was used to train the datasets and to generate the GNN embeddings. The training of BERT algorothm used Ktrain implementation with internal parameters *maxlen* and *max_features* set to 350 and 35,000 respectively.

5.4 Discussion

The comparison of performance from the proposed model involved choice of other models. LDA, NMF were chosen for their importance on the development of topic modelling and their embeddings were used to generate supervised models with Random Forest. TPBG was used for its deep reationship with the proposed model and BERT was chosen because it represents modern pre-trained models.

From the experiments it is possible to conclude that the complexity of the GNN model (represented by the number of hidden channels and convolutional layers) plays a significant role in the number of epochs the training process has execute to find the optimal solution. Better results were achieved for the more complex models.

The values for F1-score do not outperform current state-of-the-art models such as BERT but, in general, the proposed model adds a residual improvement to the TPBG classification, in which the bi-partite structure of the graph is based. BERT is a model that can retrieve contextual information from texts, representing them in a token vocabulary space. It benefits from the fact that the average number of terms in each document is smaller than the number of tokens from the model.

It is also possible to conclude that the choice of the loss function variation does not significantly affect the performance metric, but it does affect, in some scenarios, the number of epochs executed to achieve optimal solution.

5.5 Complexity Analysis

The computational complexity of the hybrid proposed solution has three components:

- PBG training: $\mathcal{O}(MTN|\mathbf{x}_W|)$
- Doc2vec features generation: $\mathcal{O}(M * log(N))$
- bi-partite graph generation: $\mathcal{O}(M|\mathbf{x}_W|)$
- GNN training: $\mathcal{O}(\prod_{k=1}^{k=K} H_k)$

where M is the number of documents, N is the vocabulary size, $|x_W|$ is the number of topic projections (the number of node features), K is the number of layers and H_k is the number of samples neighbours for each layer (limited to the number of nodes).

The number of GNN layers is required to be low (in experiments it was limited to 4) due to the over-smoothing effect of aggregating messages from node embeddings.

5.6 Summary

The experiments have shown that it is possible to combine a technique that propagates topic information through a bi-partite graph structure and use the latent topic representations as node features to feed a graph neural network based on sample message passing. The overall cost of the procedure is concentrated in the second stage of the training, due to the bigger structure of the GNN requiring more computational power than TPBG.

The text classification experimental results for this proposed model are comparable with benchmark techniques available in the literature, achieving a residual improvement when compared to TPBG.

CHAPTER

6

CONCLUSION

This work developed an analysis of the Graph Neural Networks and proposed an application combining TPBG (FALEIROS; VALEJO; LOPES, 2020), a topic model not based on deep learning with GNN model for the text classification problem.

The theoretical foundations of GNN are presented to serve as reference material on the subject with references from the main works from the literature. Regarding the proposed model, classification experiments with benchmark open source datasets show that it is capable of successfully predict the classes of texts. The model's performance was, for some scenarios, comparable with selected alternative models related to the subject.

The present work used the GraphSAGE (HAMILTON; YING; LESKOVEC, 2017) GNN model, chosen by its capability to process large graphs, such as document-words graphs. In future research it is possible to investigate the feasibility of applying combinations of other models of GNNs and how the bi-partite generated graph can bring information on the explainability, searching for connectivity patterns as well as using an arbitrary number of document classes for model training.

BIBLIOGRAPHY

AHMED, A.; SHERVASHIDZE, N.; NARAYANAMURTHY, S.; JOSIFOVSKI, V.; SMOLA, A. J. Distributed large-scale natural graph factorization. In: **Proceedings of the 22nd international conference on World Wide Web**. [S.l.: s.n.], 2013. p. 37–48. Citation on page 37.

BACCIU, D.; ERRICA, F.; MICHELI, A.; PODDA, M. A gentle introduction to deep learning for graphs. **Neural Networks**, Elsevier, v. 129, p. 203–221, 2020. Citation on page 53.

BAEZA-YATES, R.; RIBEIRO-NETO, B. *et al.* **Modern information retrieval**. [S.l.]: ACM press New York, 1999. Citation on page 55.

BALDASSARRE, F.; AZIZPOUR, H. Explainability techniques for graph convolutional networks. **arXiv preprint arXiv:1905.13686**, 2019. Citations on pages 52 and 53.

BATTAGLIA, P. W.; HAMRICK, J. B.; BAPST, V.; SANCHEZ-GONZALEZ, A.; ZAMBALDI, V.; MALINOWSKI, M.; TACCHETTI, A.; RAPOSO, D.; SANTORO, A.; FAULKNER, R. *et al.* Relational inductive biases, deep learning, and graph networks. **arXiv preprint arXiv:1806.01261**, 2018. Citation on page 53.

BELKIN, M.; NIYOGI, P. Laplacian eigenmaps and spectral techniques for embedding and clustering. **Advances in neural information processing systems**, v. 14, 2001. Citation on page 37.

BERG, R. v. d.; KIPF, T. N.; WELLING, M. Graph convolutional matrix completion. **arXiv preprint arXiv:1706.02263**, 2017. Citation on page 27.

BHAGAT, S.; CORMODE, G.; MUTHUKRISHNAN, S. Node classification in social networks. In: **Social network data analytics**. [S.l.]: Springer, 2011. p. 115–148. Citation on page 25.

BLEI, D. M.; NG, A. Y.; JORDAN, M. I. Latent dirichlet allocation. **Journal of machine Learning research**, v. 3, n. Jan, p. 993–1022, 2003. Citations on pages 56 and 57.

BRODY, S.; ALON, U.; YAHAV, E. How attentive are graph attention networks? **arXiv preprint arXiv:2105.14491**, 2021. Citation on page 49.

BRUNA, J.; ZAREMBA, W.; SZLAM, A.; LECUN, Y. Spectral networks and locally connected networks on graphs. **arXiv preprint arXiv:1312.6203**, 2013. Citation on page 44.

BUITINCK, L.; LOUPPE, G.; BLONDEL, M.; PEDREGOSA, F.; MUELLER, A.; GRISEL, O.; NICULAE, V.; PRETTENHOFER, P.; GRAMFORT, A.; GROBLER, J.; LAYTON, R.; VANDERPLAS, J.; JOLY, A.; HOLT, B.; VAROQUAUX, G. API design for machine learning software: experiences from the scikit-learn project. In: **ECML PKDD Workshop: Languages for Data Mining and Machine Learning**. [S.l.: s.n.], 2013. p. 108–122. Citation on page 66.

CAO, S.; LU, W.; XU, Q. Grarep: Learning graph representations with global structural information. In: **Proceedings of the 24th ACM international on conference on information and knowledge management**. [S.l.: s.n.], 2015. p. 891–900. Citation on page 37.

CRAVEN, M.; MCCALLUM, A.; PIPASQUO, D.; MITCHELL, T.; FREITAG, D. **Learning to extract symbolic knowledge from the World Wide Web**. [S.l.], 1998. Citation on page 66.

DAI, H.; DAI, B.; SONG, L. Discriminative embeddings of latent variable models for structured data. In: PMLR. **International conference on machine learning**. [S.l.], 2016. p. 2702–2711. Citation on page 46.

DEERWESTER, S.; DUMAIS, S. T.; FURNAS, G. W.; LANDAUER, T. K.; HARSHMAN, R. Indexing by latent semantic analysis. **Journal of the American society for information science**, Wiley Online Library, v. 41, n. 6, p. 391–407, 1990. Citation on page 56.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. **arXiv preprint arXiv:1810.04805**, 2018. Citation on page 58.

DIENG, A. B.; WANG, C.; GAO, J.; PAISLEY, J. Topicrnn: A recurrent neural network with long-range semantic dependency. **arXiv preprint arXiv:1611.01702**, 2016. Citation on page 58.

DWIVEDI, V. P.; BRESSON, X. A generalization of transformer networks to graphs. **arXiv preprint arXiv:2012.09699**, 2020. Citation on page 48.

FALEIROS, T. de P.; LOPES, A. de A. On the equivalence between algorithms for non-negative matrix factorization and latent dirichlet allocation. In: **ESANN**. [S.l.: s.n.], 2016. Citation on page 57.

FALEIROS, T. de P.; VALEJO, A.; LOPES, A. de A. Unsupervised learning of textual pattern based on propagation in bipartite graph. **Intelligent Data Analysis**, IOS Press, v. 24, n. 3, p. 543–565, 2020. Citations on pages 58, 59, 60, 63, and 83.

FEY, M.; LENSSEN, J. E. Fast graph representation learning with pytorch geometric. **arXiv preprint arXiv:1903.02428**, 2019. Citations on pages 53, 66, and 67.

FORMAN, G. **19MclassTextWc dataset**. [S.l.]: Academic Press, 2006. Citation on page 66.

GAUDELET, T.; DAY, B.; JAMASB, A. R.; SOMAN, J.; REGEP, C.; LIU, G.; HAYTER, J. B.; VICKERS, R.; ROBERTS, C.; TANG, J. *et al.* Utilizing graph machine learning within drug discovery and development. **Briefings in bioinformatics**, Oxford University Press, v. 22, n. 6, p. bbab159, 2021. Citation on page 52.

GILMER, J.; SCHOENHOLZ, S. S.; RILEY, P. F.; VINYALS, O.; DAHL, G. E. Neural message passing for quantum chemistry. In: PMLR. **International conference on machine learning**. [S.l.], 2017. p. 1263–1272. Citations on pages 38, 51, and 52.

GORI, M.; MONFARDINI, G.; SCARSELLI, F. A new model for learning in graph domains. In: IEEE. **Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.** [S.l.], 2005. v. 2, p. 729–734. Citations on pages 26 and 38.

GRAHAM, A. Dmoz—directory mozilla the open directory project, http://dmoz. org. **The Physics Teacher**, American Association of Physics Teachers, v. 42, n. 4, p. 255–255, 2004. Citation on page 66.

GRATTAROLA, D.; ALIPPI, C. Graph neural networks in tensorflow and keras with spektral [application notes]. **IEEE Computational Intelligence Magazine**, IEEE, v. 16, n. 1, p. 99–106, 2021. Citation on page 53.

GREENE, D.; CUNNINGHAM, P. Practical solutions to the problem of diagonal dominance in kernel document clustering. In: **Proc. 23rd International Conference on Machine learning (ICML'06)**. [S.l.]: ACM Press, 2006. p. 377–384. Citation on page 66.

GROVER, A.; LESKOVEC, J. node2vec: Scalable feature learning for networks. In: **Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.: s.n.], 2016. p. 855–864. Citation on page 37.

GUPTA, P.; CHAUDHARY, Y.; BUETTNER, F.; SCHÜTZE, H. Document informed neural autoregressive topic models with distributional prior. In: **Proceedings of the AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2019. v. 33, n. 01, p. 6505–6512. Citation on page 58.

HAMILTON, W. L. Graph representation learning. **Synthesis Lectures on Artificial Intelligence and Machine Learning**, Morgan & Claypool Publishers, v. 14, n. 3, p. 1–159, 2020. Citations on pages 37, 38, and 40.

HAMILTON, W. L.; YING, R.; LESKOVEC, J. Inductive representation learning on large graphs. In: **Proceedings of the 31st International Conference on Neural Information Processing Systems**. [S.l.: s.n.], 2017. p. 1025–1035. Citations on pages 27, 47, 51, 63, and 83.

HAMMOND, D. K.; VANDERGHEYNST, P.; GRIBONVAL, R. Wavelets on graphs via spectral graph theory. **Applied and Computational Harmonic Analysis**, Elsevier, v. 30, n. 2, p. 129–150, 2011. Citation on page 45.

HAYKIN, S. **Neural networks and learning machines, 3/E**. [S.l.]: Pearson Education India, 2010. Citations on pages 33 and 34.

HOFMANN, T. Probabilistic latent semantic indexing. In: **Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval**. [S.l.: s.n.], 1999. p. 50–57. Citation on page 56.

HUANG, Q.; YAMADA, M.; TIAN, Y.; SINGH, D.; CHANG, Y. Graphlime: Local interpretable model explanations for graph neural networks. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, 2022. Citation on page 53.

JOHNSON, J.; GUPTA, A.; FEI-FEI, L. Image generation from scene graphs. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2018. p. 1219–1228. Citation on page 27.

JONES, S.; CUNNINGHAM, S. J.; MCNAB, R. Usage analysis of a digital library. In: **Proceedings of the third ACM conference on Digital libraries**. [S.l.: s.n.], 1998. p. 293–294. Citation on page 66.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014. Citation on page 67.

KIPF, T. N.; WELLING, M. Semi-supervised classification with graph convolutional networks. **arXiv preprint arXiv:1609.02907**, 2016. Citations on pages 27, 45, and 51.

____. Variational graph auto-encoders. **arXiv preprint arXiv:1611.07308**, 2016. Citations on pages 39 and 51.

KULLBACK, S.; LEIBLER, R. A. On information and sufficiency. **The annals of mathematical statistics**, JSTOR, v. 22, n. 1, p. 79–86, 1951. Citation on page 46.

LAROCHELLE, H.; LAULY, S. A neural autoregressive topic model. **Advances in Neural Information Processing Systems**, v. 25, 2012. Citation on page 58.

LE, Q.; MIKOLOV, T. Distributed representations of sentences and documents. In: PMLR. **International conference on machine learning**. [S.l.], 2014. p. 1188–1196. Citation on page 63.

LI, S.; ZHOU, J.; XU, T.; HUANG, L.; WANG, F.; XIONG, H.; HUANG, W.; DOU, D.; XIONG, H. Structure-aware interactive graph neural networks for the prediction of protein-ligand binding affinity. In: **Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining**. [S.l.: s.n.], 2021. p. 975–985. Citation on page 27.

LOPES, N.; RIBEIRO, B. Non-negative matrix factorization (nmf). In: **Machine Learning for Adaptive Many-Core Machines-A Practical Approach**. [S.l.]: Springer, 2015. p. 127–154. Citation on page 57.

LUO, D.; CHENG, W.; XU, D.; YU, W.; ZONG, B.; CHEN, H.; ZHANG, X. Parameterized explainer for graph neural network. **Advances in neural information processing systems**, v. 33, p. 19620–19631, 2020. Citation on page 53.

M., D. **Classic3 and classic4 datasets**. 2010. Available: <http://www.dataminingresearch.com/index.php/2010/09/classic3-classic4-datasets/>. Citation on page 66.

MALLAT, S. **A wavelet tour of signal processing**. [S.l.]: Elsevier, 1999. Citation on page 44.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, n. 4, p. 115–133, 1943. Citation on page 32.

MITCHELL, T. M. **Machine Learning**. New York: McGraw-Hill, 1997. ISBN 978-0-07-042807-2. Citation on page 25.

OPENAI. Gpt-4 technical report. **ArXiv**, abs/2303.08774, 2023. Citation on page 58.

OU, M.; CUI, P.; PEI, J.; ZHANG, Z.; ZHU, W. Asymmetric transitivity preserving graph embedding. In: **Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.: s.n.], 2016. p. 1105–1114. Citation on page 37.

PAZZANI, M. J. **UCI Machine Learning Repository**. 2017. Available: <http://archive.ics.uci.edu/ml>. Citation on page 66.

PEROZZI, B.; AL-RFOU, R.; SKIENA, S. Deepwalk: Online learning of social representations. In: **Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining**. [S.l.: s.n.], 2014. p. 701–710. Citations on pages 26 and 37.

RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. *et al.* Improving language understanding by generative pre-training. OpenAI, 2018. Citation on page 58.

REHUREK, R.; SOJKA, P. Gensim–python framework for vector space modelling. **NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic**, v. 3, n. 2, 2011. Citation on page 67.

RENNIE, J. **20 Newsgroups Dataset**. 2008. Available: <http://people.csail.mit.edu/jrennie/20Newsgroups/>. Citation on page 66.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citation on page 32.

ROSSI, R. G.; MARCACINI, R. M.; REZENDE, S. O. Benchmarking text collections for classification and clustering tasks. 2013. Citation on page 67.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **nature**, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986. Citation on page 34.

RUMETHART, D. Learning representations by back-propagating errors. **Nature**, v. 323, p. 533–536, 1986. Citation on page 25.

SANCHEZ-GONZALEZ, A.; GODWIN, J.; PFAFF, T.; YING, R.; LESKOVEC, J.; BATTAGLIA, P. Learning to simulate complex physics with graph networks. In: PMLR. **International Conference on Machine Learning**. [S.l.], 2020. p. 8459–8468. Citation on page 27.

SANCHEZ-LENGELING, B.; WEI, J.; LEE, B.; REIF, E.; WANG, P.; QIAN, W.; MCCLOSKEY, K.; COLWELL, L.; WILTSCHKO, A. Evaluating attribution for graph neural networks. **Advances in neural information processing systems**, v. 33, p. 5898–5910, 2020. Citation on page 52.

SCARSELLI, F.; GORI, M.; TSOI, A. C.; HAGENBUCHNER, M.; MONFARDINI, G. The graph neural network model. **IEEE transactions on neural networks**, IEEE, v. 20, n. 1, p. 61–80, 2008. Citation on page 38.

SCHNAKE, T.; EBERLE, O.; LEDERER, J.; NAKAJIMA, S.; SCHÜTT, K. T.; MÜLLER, K.-R.; MONTAVON, G. Xai for graphs: explaining graph neural network predictions by identifying relevant walks. 2020. Citation on page 53.

SHEN, D.; QIN, C.; WANG, C.; DONG, Z.; ZHU, H.; XIONG, H. Topic modeling revisited: A document graph-based neural network perspective. **Advances in Neural Information Processing Systems**, v. 34, p. 14681–14693, 2021. Citation on page 62.

SHUMAN, D. I.; NARANG, S. K.; FROSSARD, P.; ORTEGA, A.; VANDERGHEYNST, P. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. **IEEE signal processing magazine**, IEEE, v. 30, n. 3, p. 83–98, 2013. Citation on page 43.

STOKES, J. M.; YANG, K.; SWANSON, K.; JIN, W.; CUBILLOS-RUIZ, A.; DONGHIA, N. M.; MACNAIR, C. R.; FRENCH, S.; CARFRAE, L. A.; BLOOM-ACKERMANN, Z. *et al.* A deep learning approach to antibiotic discovery. **Cell**, Elsevier, v. 180, n. 4, p. 688–702, 2020. Citation on page 52.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. **Advances in neural information processing systems**, v. 30, 2017. Citations on pages 39 and 48.

VELIČKOVIĆ, P.; CUCURULL, G.; CASANOVA, A.; ROMERO, A.; LIO, P.; BENGIO, Y. Graph attention networks. **arXiv preprint arXiv:1710.10903**, 2017. Citations on pages 48, 49, and 51.

VIRINCHI, S.; SALADI, A. S. V. K. K.; MONDAL, A. Recommending related products using graph neural networks in directed graphs. In: **ECML-PKDD 2022**. [s.n.], 2022. Available: <https://www.amazon.science/publications/recommending-related-products-using-graph-neural-networks-in-directed-graphs>. Citation on page 53.

VISHWANATHAN, S. V. N.; SCHRAUDOLPH, N. N.; KONDOR, R.; BORGWARDT, K. M. Graph kernels. **Journal of Machine Learning Research**, MIT Press, v. 11, p. 1201–1242, 2010. Citation on page 25.

WANG, M.; ZHENG, D.; YE, Z.; GAN, Q.; LI, M.; SONG, X.; ZHOU, J.; MA, C.; YU, L.; GAI, Y. *et al.* Deep graph library: A graph-centric, highly-performant package for graph neural networks. **arXiv preprint arXiv:1909.01315**, 2019. Citation on page 53.

WANG, Y.; SUN, Y.; LIU, Z.; SARMA, S. E.; BRONSTEIN, M. M.; SOLOMON, J. M. Dynamic graph cnn for learning on point clouds. **Acm Transactions On Graphics (tog)**, ACM New York, NY, USA, v. 38, n. 5, p. 1–12, 2019. Citation on page 27.

WEISFEILER, B.; LEMAN, A. The reduction of a graph to canonical form and the algebra which appears therein. **NTI, Series**, v. 2, n. 9, p. 12–16, 1968. Citation on page 46.

WU, L.; CUI, P.; PEI, J.; ZHAO, L.; SONG, L. Graph neural networks. In: **Graph Neural Networks: Foundations, Frontiers, and Applications**. [S.l.]: Springer, 2022. p. 27–37. Citation on page 51.

WU, Z.; PAN, S.; CHEN, F.; LONG, G.; ZHANG, C.; PHILIP, S. Y. A comprehensive survey on graph neural networks. **IEEE transactions on neural networks and learning systems**, IEEE, v. 32, n. 1, p. 4–24, 2020. Citation on page 26.

XU, D.; ZHU, Y.; CHOY, C. B.; FEI-FEI, L. Scene graph generation by iterative message passing. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2017. p. 5410–5419. Citation on page 27.

XU, K.; HU, W.; LESKOVEC, J.; JEGELKA, S. How powerful are graph neural networks? **arXiv preprint arXiv:1810.00826**, 2018. Citations on pages 47 and 52.

XUE, N. Steven bird, evan klein and edward loper. natural language processing with python. o'reilly media, inc. 2009. isbn: 978-0-596-51649-9. **Natural Language Engineering**, Cambridge University Press, v. 17, n. 3, p. 419–424, 2011. Citation on page 66.

YANG, L.; WU, F.; GU, J.; WANG, C.; CAO, X.; JIN, D.; GUO, Y. Graph attention topic modeling network. In: **Proceedings of The Web Conference 2020**. [S.l.: s.n.], 2020. p. 144–154. Citations on pages 27 and 61.

YAO, H.; WU, F.; KE, J.; TANG, X.; JIA, Y.; LU, S.; GONG, P.; YE, J.; LI, Z. Deep multi-view spatial-temporal network for taxi demand prediction. In: **Proceedings of the AAAI Conference on Artificial Intelligence**. [S.l.: s.n.], 2018. v. 32, n. 1. Citation on page 27.

YAO, L.; MAO, C.; LUO, Y. Graph convolutional networks for text classification. In: **Proceedings of the AAAI conference on artificial intelligence**. [S.l.: s.n.], 2019. v. 33, n. 01, p. 7370–7377. Citation on page 27.

YING, R.; HE, R.; CHEN, K.; EKSOMBATCHAI, P.; HAMILTON, W. L.; LESKOVEC, J. Graph convolutional neural networks for web-scale recommender systems. In: **Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining**. [S.l.: s.n.], 2018. p. 974–983. Citation on page 27.

YING, Z.; BOURGEOIS, D.; YOU, J.; ZITNIK, M.; LESKOVEC, J. Gnnexplainer: Generating explanations for graph neural networks. **Advances in neural information processing systems**, v. 32, 2019. Citation on page 53.

YUAN, H.; TANG, J.; HU, X.; JI, S. Xgnn: Towards model-level explanations of graph neural networks. In: **Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining**. [S.l.: s.n.], 2020. p. 430–438. Citation on page 53.

YUAN, H.; YU, H.; GUI, S.; JI, S. Explainability in graph neural networks: A taxonomic survey. **arXiv preprint arXiv:2012.15445**, 2020. Citation on page 52.

YUN, S.; JEONG, M.; KIM, R.; KANG, J.; KIM, H. J. Graph transformer networks. **Advances in neural information processing systems**, v. 32, 2019. Citation on page 49.

ZAHEER, M.; KOTTUR, S.; RAVANBAKHSH, S.; POCZOS, B.; SALAKHUTDINOV, R. R.; SMOLA, A. J. Deep sets. **Advances in neural information processing systems**, v. 30, 2017. Citation on page 39.

ZHANG, J.; SHI, X.; XIE, J.; MA, H.; KING, I.; YEUNG, D.-Y. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. **arXiv preprint arXiv:1803.07294**, 2018. Citation on page 27.

ZHANG, M.; CHEN, Y. Link prediction based on graph neural networks. **Advances in neural information processing systems**, v. 31, 2018. Citation on page 52.

ZHANG, X.; HE, Y.; BRUGNONE, N.; PERLMUTTER, M.; HIRN, M. Magnet: A neural network for directed graphs. **Advances in neural information processing systems**, v. 34, p. 27003–27015, 2021. Citation on page 62.

ZHANG, Y.; DEFAZIO, D.; RAMESH, A. Relex: A model-agnostic relational model explainer. In: **Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society**. [S.l.: s.n.], 2021. p. 1042–1049. Citation on page 53.

ZHOU, D.; HU, X.; WANG, R. Neural topic modeling by incorporating document relationship graph. **arXiv preprint arXiv:2009.13972**, 2020. Citations on pages 27, 60, and 61.

ZHOU, J.; CUI, G.; HU, S.; ZHANG, Z.; YANG, C.; LIU, Z.; WANG, L.; LI, C.; SUN, M. Graph neural networks: A review of methods and applications. **AI Open**, Elsevier, v. 1, p. 57–81, 2020. Citation on page 41.

ZHU, R.; ZHAO, K.; YANG, H.; LIN, W.; ZHOU, C.; AI, B.; LI, Y.; ZHOU, J. Aligraph: a comprehensive graph neural network platform. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 12, n. 12, p. 2094–2105, 2019. Citation on page 53.


ICMC
USP
SÃO CARLOS