Improving the Interpretability of Machine Learning Approaches with User-generated Data

By

Giuseppe Serra

A thesis submitted to
the University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

Centre of Excellence for Research in Computational Intelligence and Applications
School of Computer Science
College of Engineering and Physical Sciences
University of Birmingham
December 2022

UNIVERSITY OF
BIRMINGHAM

University of Birmingham Research Archive

e-theses repository

This unpublished thesis/dissertation is copyright of the author and/or third parties. The intellectual property rights of the author or third parties in respect of this work are as defined by The Copyright Designs and Patents Act 1988 or as modified by any successor legislation.

Any use made of information contained in this thesis/dissertation must be in accordance with that legislation and must be properly acknowledged. Further distribution or reproduction in any format is prohibited without the permission of the copyright holder.

© Copyright by GIUSEPPE SERRA, 2022

All Rights Reserved

ABSTRACT

Given the increasing deployment of automatic decision systems in many critical scenarios, understanding and explaining machine-based decision systems has become an important problem to be solved. For this reason, recently, there has been an increasing interest in what we commonly call Explainable AI (XAI). In this thesis, starting from an overview of the inner mechanisms of deep neural networks, we will tackle the problem of interpretability from two different perspectives.

First, considering that neural networks usually encode the input features as numerical vector representations hardly understandable by humans, we propose new approaches to learning interpretable numerical vectors. In view of the availability of large collections of textual data in different scenarios, we intend to exploit the natural language information to generate vectors with intrinsic interpretability. In this way, the new numerical vectors will have the capacity of being effectively used by neural algorithms while also providing human-understandable information. The proposed methodologies are evaluated on e-commerce data with textual reviews. In this context, given the so-called *neural hype*, we also critically analyze whether the use of complicated and deep architectures is fully motivated in recommender systems scenarios.

Second, given the inscrutability of the inner reasoning of neural architectures, we develop a new approach capable of highlighting the portion of input information effectively used by the model for its predictions. The methodology is based on the so-called *learning to explain* paradigm and is applied to graph-based models. The proposed method learns to select subgraphs that are used in all the computational operations until prediction. The inherent interpretability of the model overcomes the limitations of common post-hoc explanation techniques. Furthermore, since the resulting explanations are faithful to the model reasoning, the results can also be used for model debugging and hyperparameter tuning.

DEDICATION

Dedicated to my parents.

ACKNOWLEDGMENTS

Words can never be enough to express my gratitude towards my supervisors — Peter Tiňo, Zhao Xu, and Xin Yao — for their endless support and guidance during this journey. I learned a lot from each piece of advice, each comment, and each challenge they provided me. I hope that, in the future, I will be able to carry on their truly inspiring lessons and become a researcher they would be proud of. I wish to thank my former colleagues in the Machine Learning group at NEC Labs. Every small chat, discussion, meeting, or presentation, was a small step to grow in many professional and personal ways. In particular, I would like to thank Mathias Niepert and Carolin Lawrence. I was lucky enough to collaborate with them, and they are a brilliant example of what it means to be an excellent researcher. Last but not least, I would also like to thank all my colleagues involved in the ECOLE project. The pandemic deprived us of many occasions for physically meeting, but I will always treasure the valuable experience we shared.

Finally, some personal acknowledgment. I want to thank my girlfriend for always supporting me through thick and thin. My closest friends for our strong bonding despite the distance. My parents for their unconditioned love and encouragement. Nothing would have been possible without the fondness of each of them.

Fundings. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme ECOLE (grant agreement No. 766186).

Gutta cavat lapidem

[non vi sed saepe cadendo]

Ovidio, "Epistulae ex Ponto"

Contents

List of Figures

List of Tables

Acronyms

a.k.a. also called as.

AI Artificial Intelligence.

c.d.f. cumulative distribution function.

CF Collaborative Filtering.

CNN Convolutional Neural Network.

DLGM Deep Latent Generative Model.

DNN Deep Neural Network.

e.g. Exempli Gratia.

ELBO Evidence Lower BOund.

EM Expectation-Maximization.

FCN Fully Connected Network.

GNN Graph Neural Network.

i.e. Id Est.

i.i.d. independent and identical distributed.

iGNN Interpretable Graph Neural Network.

KL Kullback-Leibler.

L2X Learning to Explain.

MC Monte Carlo.

ML Machine Learning.

MLE Maximum Likelihood Estimation.

MLP Multi Layer Perceptron.

MSE Mean Squared Error.

NLP Natural Language Programming.

NN Neural Network.

p.m.f. probability mass function.

ReLU Rectified Linear Units.

SDLGM Sparse Deep Latent Generative Model.

SGD Stochastic Gradient Descent.

SOM Self-Organizing Maps.

VI Variational Inference.

XAI Explainable AI.

Chapter One

Introduction

1.1 Motivations

The impressive advance of machine learning (ML) during the past few years has shifted the priorities and goals related to ML approaches. In the past, the main objective was to improve the predictive capabilities of the models. Afterward, thanks to modern technological advancement, we were able to progressively build more powerful and resourceful machines allowing, in parallel, steady progress of the learning capabilities of the predictive models. Nowadays, given the impressive predictive performance of neural-based approaches, it is not particularly complicated to achieve competitive results on many tasks. Therefore, researchers have started focusing on other challenges and problems related to neural-based models. One of the main challenges is related to the *black-box* nature of the models and their complexity. In fact, along with the improvements in predictive accuracy, also the complexity of the architectures is increased substantially. Most of the breakthrough architectures [1]–[4] recently proposed on several domains correspond to models with billion parameters. Thus, given the continuous more use of ML models in real-world applications, there is an increasing need for understanding the rationales of these decision systems since, although having tremendous

predictive capabilities, there is little comprehension from the human perspective. While this is entirely acceptable for low-risk scenarios (e.g., movie recommendation), the same does not hold when we want to integrate these approaches into decision-critical systems. The ubiquitous adoption of computer-aided frameworks in critical tasks requires fully understanding the models before deploying them in the real world. For this reason, there has been a surge in research regarding Explainable AI (XAI) and its related aspects.

1.2 Contributions and Scope

Considering the behavior and the learning process of neural networks (NNs), we can face the explainability problem from different perspectives. Neural networks, for efficient computation, usually transform the input data into some latent numerical representations, also known as embeddings. Vector space modeling has become increasingly more popular since the introduction of Word2Vec [5] in 2013 for Natural Language Processing (NLP) tasks. It originally consisted of representing (i.e., embedding) text tokens in a continuous vector space where similar items were mapped to closer low-dimensional vector representations in the embedding space. In this way, semantically similar concepts were supposed to be close to each other in the vector space allowing the automatic extraction of knowledge from the input data that would be difficult to discover otherwise.

The idea of automatically learning effective latent representations of the input data was advantageous and, given the success of Word2Vec, embedding learning has become a well-established and common procedure in many applications. For example, in graph-based applications, we have *node embedding* or *graph embedding* learning. In the first case, we aim to encode the nodes as low-dimensional vectors that reflect their closeness, relationships, or similarities. In this way, for instance, nodes can be projected into a latent space such that

their geometric relations in the embedding space correspond to some type of similarity in the original network [6], [7]. Similarly, graph embedding consists of learning low-dimensional representations of entire graphs such that the ones sharing similar properties or structures in the original space lie close to each other in the embedding space. A graphical example of node embedding is provided in Figure 1.1.

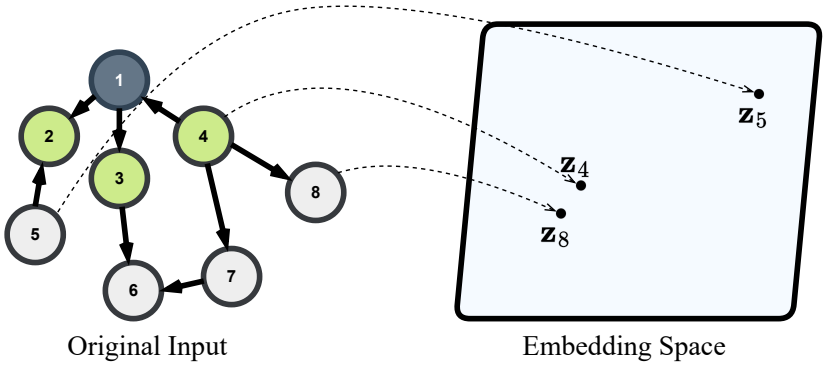


Figure 1.1: An illustration of node embedding learning for graph data.

The dense low-dimensional representations learned by generic embedding methods, although having a meaning in the latent space, are not easily understandable from the human perspective. Differently from human-engineered feature vectors, where we exactly know the feature associated with each dimension of a vector, we are not able to decipher the meaning of the numerical values associated with a vector generated by the embedding layer of a model.

$$\mathbf{z}_4 = [0.43, 0.58, 0.91, 0.01, 0.62, 0.05]$$

$$\mathbf{z}_5 = [0.12, 0.23, 0.56, 0.84, 0.65, 0.72]$$

$$\mathbf{z}_8 = [0.47, 0.64, 0.86, 0.10, 0.75, 0.19]$$

For instance, if we consider the analysis of the vectors above, one can see their relationships when projected into a 2-dimensional space (see Figure 1.1) but is not able to gather any valuable information from the corresponding numbers since no human-understandable concept is associated with them. Therefore, by not knowing the meaning of each dimension of the learned embedding vectors, the evaluation of a single vector is not possible. In this direction,

following the aforementioned limitation, we will focus on proposing new methodologies that would help organize these internal representations in a meaningful way. Our goal is to generate vector representations that can be evaluated both collectively and singularly. A simple but effective idea is to transform the vector dimensions into features whose meaning can be understood by humans. In this way, high or low values of certain features suggest what the model thinks is relevant for the prediction and, in turn, can serve as a human-understandable explanation of the internal behavior of the models.

Another problem related to neural-based approaches is their opaque nature. In fact, neural networks are considered black-boxes for which we know the input and the output, but for which we are not able to inspect and evaluate their internal reasoning process.

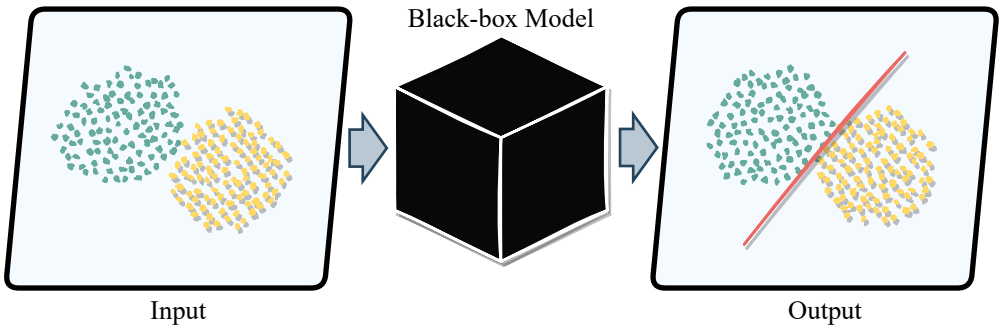


Figure 1.2: A graphical illustration of a learning system involving a black-box model.

Since we are unable to explain why a certain prediction has been made, this may be a critical problem when we employ black-box models in critical tasks. As reported in [8], it has been shown that the use of non-interpretable machine learning models in real-world applications has had serious consequences in finance, justice, and many other domains. The issue can be solved following two different directions: 1) by using intrinsic interpretable models (e.g., regression models, decision trees); 2) by implementing methodologies that would open black-box models in order to understand their internal reasoning.

For the first case, despite the limited expressiveness compared to neural networks, it has

been shown that the use of complicated and difficult-to-interpret models is not needed in many real tasks [9], [10]. Following the critical examination of the *neural-hype*, we will try to understand whether we really need to implement neural-based approaches or, contrarily, we can make advantage of simpler and easy-to-interpret methodologies for the same tasks.

For the second case, instead, we aim to explain the reasoning of the neural architectures in order to increase their clearness and, in turn, human trust. Furthermore, understanding the behavior of the models brings other advantages. For instance, we can analyze whether the model is focusing on the expected features or, on the contrary, it is affected by some learning problem such as *shortcut learning* [11]. Thus, the analysis of the inner mechanisms can facilitate model analysis and debug before the deployment of the model in the real world.

Sometimes, the additional information related to our datasets can further help us in improving the interpretability of the results. Since textual information is one of the most natural ways to explain concepts to humans, throughout this thesis, whenever available, we will try to make use of the textual information for improving the interpretability of the results.

Challenges. Instilling interpretability within learning methods brings additional challenges. For instance, there might be a reduction in the predictive performance of the model. This is motivated by the fact that interpretability constraints could limit the expressiveness of the learning process. Considering the embedding problem, the definition of specific properties of the values and features associated with the latent vectors could limit their discriminative power degrading the predictive capacity of the models. Similarly, if we try to extract a subset of the input features that contribute most to the prediction, by reducing the information used by the model for inference, we may have a drop in its predictive ability. An additional challenge, given the ideal properties of an explanation, is how to quantify the quality of an interpretation without human intervention. On this matter, there is no consensus regarding a common metric nor regarding the definition of what *'explainability'*

exactly means. In a more general perspective, in this thesis, we will consider explainable any method, approach, or mechanism that could help humans increase their trust and comprehension of machine learning models. In particular, our intent is to provide an understanding of the input features considered relevant by the model for making its predictions. Differently from post-hoc techniques, we aim to generate explanations that faithfully reflect the model reasoning and can serve as an interpretation of the model outcome. In terms of the human-readability of the explanations, the goal is to provide explanations that can deliver direct feedback to all the end-users by considering several aspects such as conciseness, consistency, and faithfulness.

Research Questions. Considering the facets and challenges of XAI just presented, the main research questions that we will try to answer are the followings:

RQ1 *Trade-off Explainability/Performance* - Does the interpretability constraint affect the learning task performance?

RQ2 *Quantifying Explainability* - Can we introduce some metrics to help assessing the quality of the explanatory results without human intervention?

RQ3 *Neural Hype and Illusion of Progress* - Are deep models and complex representations really better than more principled and easy-to-interpret approaches?

Summarizing, throughout this thesis, we focus on introducing and proposing new approaches that could enhance the interpretability of machine learning approaches. The problem is faced from different perspectives including a) the improvement of latent representation understandability, and b) the comprehension of the inner reasoning of black-box models. Depending on the task, when possible, we will also make use of the extra human-understandable information which could further help us in creating explanations easily accessible not only by researchers but also by non-expert end-users.

1.3 Outline

As just said, in the remainder of this thesis we will present novel approaches to improve the interpretability of black-box models, their internal representations, and the corresponding outputs.

In Chapter 3, we present a novel method for organizing the latent vector representations learned by neural networks in a meaningful way. The idea is to exploit the additional textual data available in many domains (e.g., reviews in recommendation systems, medical narratives in medical applications, or social media posts in social networks) as a human-understandable source of information to create vector features that perform competitively in a given prediction task and whose meaning can be understood by humans (**RQ1**). The work is analyzed in the context of review data. In a few words, for each user and product, we aim to generate a word-based vector where each dimension of the latent representation represents a word contained in a selected vocabulary. The corresponding value represents the probability that the given word explains user preferences or product characteristics. Furthermore, we also propose a quantitative evaluation of the generated text-based vectors in order to further ease the evaluation of the results (**RQ2**).

In Chapter 4, considering the so-called *neural-hype*, we want to understand whether we really need to implement neural (and deep) approaches and employ dense and high-dimensional embeddings for common predictive tasks, e.g., rating prediction. For this purpose, similarly to what is presented in Chapter 3, we use the textual information for organizing the user and product vectors in the latent space. However, differently from the previous approach, we implement a fully transparent probabilistic method that creates and uses interpretable, compact and rather sparse representations for rating prediction tasks (**RQ3**). We compare the results with popular neural-based approaches and we critically analyze the trade-off between the predictive capability and interpretability of the considered models (**RQ1**).

Finally, in Chapter 5, we change perspective and try to uncover the inner mechanisms of black-box models. In some cases, human-intelligible information is not available along with the data. Therefore, we need to find another way to provide explanations for the output. Instead of focusing on explaining the internal latent representations through text-based concepts, we propose a method to understand which information is used to make a certain prediction. The proposed method takes inspiration from the learning to explain (L2X) paradigm [12] and is applied to graph-based models. Through an approach based on gradient-based implicit differentiation, we aim to identify the discriminative subset of features that is effectively used by the model for the prediction (**RQ1**). Given the properties of the generated output, since it faithfully reflects the information used to make the predictions, we can analyze whether the model is focusing on the expected features or if it is affected by some learning problem (**RQ2**). Hence, by fully understanding the model reasoning, we can analyze and debug the model effectively.

Related Publications. The work presented in Chapter 3 has been published as:

- Giuseppe Serra, Zhao Xu, Mathias Niepert, Carolin Lawrence, Peter Tiňo, and Xin Yao. "Interpreting Node Embedding with Text-labeled Graphs." In 2021 International Joint Conference on Neural Networks (IJCNN), pp. 1-8. IEEE, 2021 [13].

The work presented in Chapter 4 reports unpublished work, currently under review as a journal article:

- Giuseppe Serra, Peter Tiňo, Zhao Xu, Xin Yao, "An Interpretable Alternative to Neural Representation Learning for Rating Prediction – Transparent Latent Class Modeling of User Reviews"

The work presented in Chapter 5 reports unpublished work, currently under review as a conference paper. A preprint of the work has been published as:

- Giuseppe Serra and Mathias Niepert, "L2XGNN: Learning to Explain Graph Neural Networks", arXiv preprint arXiv:2209.14402, 2022 [14].

Additional discussions and perspectives throughout the thesis are taken from other publications or contributed articles. The ideas and claims of these works can be integrated as additional features to improve the quality of the proposed approaches, but are not the core topic of the thesis. In particular, some related discussions are inspired by concepts taken from the following publications:

- Zhao Xu, Daniel Onoro-Rubio, Giuseppe Serra, and Mathias Niepert. "Learning Sparsity of Representations with Discrete Latent Variables." In 2021 International Joint Conference on Neural Networks (IJCNN), pp. 1-9. IEEE, 2021 [15]

- Bhushan Kotnis, Kiril Gashteovski, Julia Gastinger, Giuseppe Serra, Francesco Alesiani, Timo Sztyler, Shaker Ammar, Na Gong, Carolin Lawrence, Zhao Xu. "Human-Centric Research for NLP: Towards a Definition and Guiding Questions." In HCI+NLP Workshop at 2022 Annual Conference of the North American Chapter of the Association for Computational Linguistics (random author order, NAACL) [16]

Chapter Two

Background & Related Works

2.1 Graph Neural Networks

In this section, we will review the basic concepts of graph theory before digging into a more detailed description of methodologies for graph-based machine learning. After the introductory part, taken from different sources [17], [18], we will introduce the concept of message-passing neural networks, which is of fundamental importance in the development of the most popular graph neural networks (GNNs) techniques.

2.1.1 Basics of Graphs

Definition 2.1.1 (**Graph**)**.** A graph $\mathcal{G} = (V, E)$ consists of two non-empty sets V and E. V is the set of elements called vertices (or nodes), while E contains the set of unordered pairs of adjacent elements of V called edges. More formally, two vertices u, v are defined adjacent if the edge $e = (u, v) \in E$. Sometimes, especially in machine learning scenarios, nodes and edges are associated with some additional features. Therefore, we might have additional data that contains such information in form of matrices. Let denote with $n_g = |V|$ and $e_g = |E|$

the number of nodes and edges in the graph respectively. We will call $\mathbf{X} \in \mathbb{R}^{n_g \times d_n}$ the feature matrix that associates each node of the graph with a d_n-dimensional feature vector, and, similarly, with $\mathbf{E} \in \mathbb{R}^{e_g \times d_e}$ the feature matrix associated with each edge e in $\mathcal{G}$.

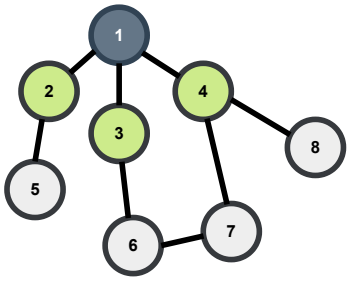


Figure 2.1: A graphical representation of a graph with 8 vertices and 8 edges.

Definition 2.1.2 (**Adjacency Matrix**)**.** The adjacency matrix $\mathbf{A} \in \{0,1\}^{n_g \times n_g}$ summarizes the information about adjacent vertices of a graph in a more convenient way. Each entry $\mathbf{A}_{uv}$ is equal to 1 if the nodes are adjacent (i.e., there exists an edge (u, v)), and 0 otherwise. Depending on the directedness of the graph, the adjacency matrix can be symmetric (undirected graph) or not (directed graph). Below, is the adjacency matrix of the graph sketched in Figure 2.1.

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Definition 2.1.3 (**Directed/Undirected Graph**)**.** A graph $\mathcal{G}$ is directed when E is a set of *ordered* pairs of nodes. In this case, each edge (u, v) has a source node u and a target node v which define the direction of the edge. Intuitively, when dealing with undirected graphs, (u, v) and (v, u) are interchangeable. In the directed case, instead, $(u, v) \neq (v, u)$. Consequently, they represent two distinct edges and may not exist simultaneously. In real-world applications, we can use undirected graphs to represent, to name a few, biological

networks (e.g., molecules) and social networks (e.g., Facebook) where the connections are non-directional. Instead, directed graphs can be used to model directional social networks (like Twitter), road maps or electrical circuits. From now on, we will consider and analyze undirected graphs. Thus, whenever we consider an edge (u, v), we are considering the edge in both directions (u, v) and (v, u). Consequently, for the definition given before, the corresponding adjacency matrix $\mathbf{A}$ is symmetrical.

Figure 2.2: Example of directed (left) and undirected (right) graph.

Definition 2.1.4 (**Connected/Disconnected Graph**)**.** An undirected graph $\mathcal{G}$ is connected if there exists a path between each pair of vertices. In other words, this means that the graph consists of a single connected component where each node can be reached starting from another vertex by means of the existing edges. Alternatively, when we have more than one connected component, the graph is said disconnected.

Figure 2.3: Example of connected (left) and disconnected (right) graph.

Definition 2.1.5 (**Graph Motif**)**.** A motif is a recurrent and statistically significant subgraph pattern that repeats itself many times among various graphs. For example, for chemical networks, a motif might be a specific chemical group that always leads to a specific

category of compounds (e.g., aromatic/non-aromatic compounds). Hence, the notion of motif is particularly important when we want to discover relevant patterns on larger graphs. As we will see in Chapter 5, by discovering discriminative subgraph motifs used by an ML model to differentiate and recognize different graph classes, we can improve the interpretability of both the results and the model itself.

2.1.2 Message-passing Graph Neural Networks

Although many GNNs exist, the majority of them can be categorized as *message-passing* graph neural networks [19]. Let $\mathcal{G}(V, E)$ be a graph with $n = |V|$ the number of nodes. Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be the feature matrix that associates each node of the graph with a d-dimensional feature vector and let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be the adjacency matrix. GNNs have three computations based on the message passing paradigm [20] which is defined as

$$\mathbf{h}_i^{\ell} = \gamma\left(\mathbf{h}_i^{\ell-1}, \square_{j \in \mathcal{N}(v_i)} \phi\left(\mathbf{h}_i^{\ell-1}, \mathbf{h}_j^{\ell-1}, r_{ij}\right)\right), \tag{2.1}$$

where γ, $\square$, and ϕ represent update, aggregation and message function respectively.

Propagation step. The message-passing network computes a message $m_{ij}^{\ell} = \phi(\mathbf{h}_i^{\ell-1}, \mathbf{h}_j^{\ell-1}, r_{ij})$ between every pair of nodes (v_i, v_j). The function takes in input v_i's and v_j's representations $\mathbf{h}_i^{\ell-1}$ and $\mathbf{h}_j^{\ell-1}$ at the previous layer $\ell - 1$, and the relation r_{ij} between the two nodes.

Aggregation step. For each node in the graph, the network performs an aggregation computation over the messages from v_i's neighborhood $\mathcal{N}(v_i)$ to calculate an aggregated message $M_i^{\ell} = \square(\{m_{ij}^{\ell} \mid v_j \in \mathcal{N}(v_i)\})$. The definition of the aggregation function differs between methods [20]–[23].

Update step. Finally, the model non-linearly transforms the aggregated message M_i^{ℓ} and v_i's representation from previous layer $\mathbf{h}_i^{\ell-1}$ to obtain v_i's representation at layer ℓ as $\mathbf{h}_i^{\ell} = \gamma(\mathbf{h}_i^{\ell-1}, M_i^{\ell})$. The final embedding for node v_i after L layers is $\mathbf{z}_i = \mathbf{h}_i^{L}$ and is used for node classification tasks. For graph classification, an additional readout function aggregates

the node representations to obtain a graph representation $\mathbf{h}_G$. This function can be any permutation invariant function or a graph-level pooling function [24]–[29]. Many aggregation schemes have been proposed lately [20]–[22], [30]–[32], however, in Chapter 5, we will make use of three particular neighborhood aggregation schemes, namely Graph Isomorphism Networks (GINs) [22], Graph Convolutional Networks (GCNs) [30], and GraphSAGE [20]. The choice is motivated by the fact that, following a general trend in GNN explainability, the explainers are usually tested on GNNs with a convolutional flavour [33]. For Graph Isomorphism Networks, the message passing operation for node v_i is

$$\mathbf{h}_i^{\ell} = \gamma^{\ell}\left(\left(1+\epsilon^{\ell}\right)\cdot \mathbf{h}_i^{\ell-1} + \sum_{j\in\mathcal{N}(v_i)} \mathbf{h}_j^{\ell-1}\right), \tag{2.2}$$

where γ represents a multi-layer perceptron (MLP), and ϵ denotes a learnable parameter. For Graph Convolutional Networks, instead, the message passing operation is defined as

$$\mathbf{h}_i^{\ell} = \gamma^{\ell}\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{h}_i^{\ell-1}\mathbf{W}^{\ell-1}\right), \tag{2.3}$$

where γ represents an activation function, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$ represents the adjacency matrix $\mathbf{A}$ with added self-loops and $\mathbf{I}_n$ the identity matrix, $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}_{ij}}$ the diagonal degree matrix, and $\mathbf{W}^{\ell-1}$ a learnable weight matrix at layer $\ell - 1$.

Finally, for GraphSAGE, the message passing operation is defined as

$$\mathbf{h}_i^{\ell} = \gamma^{\ell}\left(\mathbf{W}^{\ell-1}\cdot f\left(\mathbf{h}_i^{\ell-1}, \left\{\mathbf{h}_j^{\ell-1}\forall j\in\mathcal{N}(v_i)\right\}\right)\right), \tag{2.4}$$

where γ represents a non-linear activation function, $\mathbf{W}^{\ell-1}$ represents a weight matrix, and f represents an aggregation function such as Sum, Max, or Mean.

We will write $\mathbf{H}_{\ell} = \textsc{Gnn}_{\ell}(\boldsymbol{A}, \mathbf{H}_{\ell-1})$ as a shorthand for the application of the ℓ^{th} layer of the GNN under consideration.

2.2 Probabilistic Modeling

In this section, we provide a refresh of probability concepts used throughout the methodological chapters. A more detailed description of the theory presented in this part can be found in the following books [34], [35].

2.2.1 Basic Definitions

Definition 2.2.1 (**Probability Space**)**.** In probability theory, a *probability space* of a random process (or experiment) is fully defined by the triple $(\Omega, \mathcal{A}, P)$. The *sample space* Ω defines the set of all possible outcomes of the experiment. The *event space* $\mathcal{A} \subseteq \Omega$ is the set of events we may consider. The function $P(\cdot)$ represents a *probability measure* which assigns a *probability* for each event in the event space.

Definition 2.2.2 (σ**-algebra**)**.** The *event space* $\mathcal{A}$ is required to be a σ-algebra. More specifically, $\mathcal{A}$ is a σ-algebra if:

1. $\emptyset \in \mathcal{A}$
2. $\forall A \in \mathcal{A} \Longrightarrow (\Omega / A) \in \mathcal{A}$ (closed under complements)
3. $\forall \{A_n\}_{n\in\mathbb{N}} \Longrightarrow \bigcup_{r\in\mathbb{N}} A_r \in \mathcal{A}$ (closed under countable unions)

As a consequence of these properties, we also have the following ones:

4. $\Omega \in \mathcal{A}$
5. $\forall \{A_n\}_{n\in\mathbb{N}} \Longrightarrow \bigcap_{r\in\mathbb{N}} A_r \in \mathcal{A}$ (closed under countable intersections)

Definition 2.2.3 (**Probability Measure**)**.** A probability measure P is a *measure* on $(\Omega, \mathcal{A})$ which assigns to each event $A \in \mathcal{A}$, a *probability* $P(A)$ such that:

1. $P(A) \geq 0 \quad \forall A \in \mathcal{A}$ (non-negativity)
2. $P(\emptyset) = 0$
3. $P(\bigcup_{r \in \mathbb{N}} A_r) = \sum_{r \in \mathbb{N}} P(A_r)$ (σ-additivity)
4. $P(\Omega) = 1$

The first three properties generally define a *measure*. The last one is specifically tailored for defining the probability measure. A closely related concept is called *random variable* which is a function that maps a possible outcome of the event space to a measurable space.

Definition 2.2.4 (**Random Variable**)**.** A random variable X is a measurable function $X : \Omega \to \mathbb{R}$ which maps an outcome $\omega \in \Omega$ to a measurable space $E \subseteq \mathbb{R}$. More generally, we may be interested in computing a probability for a group B of points $\omega_i \in \Omega$ (for example, an interval). In order to compute $P(X \in B)$, X should be measurable, i.e., $X^{-1}(B) \in \mathcal{A}$. From this, we can infer that:

$$P(X \in B) = P(X(\omega) \in B) = P(\{\omega \in \Omega : X(\omega) \in B\}) = P(X^{-1}(B)). \tag{2.5}$$

Definition 2.2.5 (**Cumulative Distribution Function**)**.** In order to identify the probability distribution related to a random variable X, we can use the cumulative distribution function (c.d.f.) $F_X(x)$ defined as

$$F_X(x) = P(X \leq x) \qquad x \in \mathbb{R}. \tag{2.6}$$

Associated with a c.d.f. there is another function that, depending on the discreteness of the random variable, we call in different ways. When the random variable is discrete, that is, when it can take values from a countably infinite set S such that $P(X \in S) = 1$, we consider the *probability mass function* (p.m.f.) defined as $p(x_r) = P(X = x_r)$. For continuous random variables, instead, we define the probability density function (p.d.f.) as $f_X(x) = \frac{\partial}{\partial x} F_X(x)$. We can exploit these two definitions to obtain the probability $P(X \in B) = \sum_{x_r \in B} p(x_r)$ for

the discrete case, and $P(X \in B) = \int_{x \in B} f_X(x)dx$ in the continuous case. The set of events x_i of a random variable X and the corresponding probabilities $P(X = x_i)$ fully define a probability distribution.

For each random variable, in order to obtain a probability distribution, the necessary and sufficient conditions are:

$$P(X = x) \geq 0; \qquad \sum_x P(X = x) = 1 \qquad \text{(discrete case)}, \tag{2.7}$$

$$f_X(x) \geq 0 : \qquad \int_{-\infty}^{\infty} f_X(x) = 1 \qquad \text{(continuous case)}. \tag{2.8}$$

In most of the cases, however, we are interested in more than a single event. For multidimensional random variables, the c.d.f. is

$$F_{X_1,\ldots,X_k}(x_1, \ldots, x_k) = P(X_1 \leq x_1, \ldots, X_k \leq x_k) = P([X_1 \leq x_1] \cap \ldots \cap [X_k \leq x_k]). \tag{2.9}$$

(For the rest of the section, we will refer to the discrete case. The definitions are valid for the continuous case by changing the notation accordingly.)

The corresponding joint probability distribution is denoted by $P(X_1 = x_1, \ldots, X_k = x_k)$. Then, if the set of random variables is independent, the joint probability is given by $\prod_{i=1}^{k} P(X_i = x_i)$. Otherwise, when an event $Y = y$ *conditions* the realization of the collection of random variables X_i, we have $\prod_{i=1}^{k} P(X_i = x_i|Y = y)$.

Definition 2.2.6 (**Bayes' Rule**)**.** Given an event E and a finite set of non-compatible events A_i, if $E \subset \bigcup_i A_i$ and $P(E) \neq 0$, then

$$P(A_i|E) = \frac{P(A_i)P(E|A_i)}{P(E)} = \frac{P(A_i)P(E|A_i)}{\sum_j P(A_j)P(E|A_j)}, \tag{2.10}$$

where $P(A_i)$, called prior probability, represents the probability we assume before the realization of a certain event E. The posterior distribution $P(A_i|E)$, instead, constitutes the probability of A_j when E occurred. The equation shows how the realization of E modifies the probability of A_i from $P(A_i)$ to $P(A_i|E)$ through $P(E|A_i)$ which is the so-called likelihood.

When dealing with two-dimensional random variables (X, Y), we might be interested in getting information about the probability distribution of one of the two random variables. This process is called *marginalization* and the result is called marginal distribution.

Definition 2.2.7 (**Marginal Distribution**)**.** Given two random variables (X, Y), the marginal distributions of X and Y are given by:

$$p_{x\cdot} = P(X = x) = \sum\nolimits_y P(X = x, Y = y) \tag{2.11}$$

$$p_{\cdot y} = P(Y = y) = \sum\nolimits_x P(X = x, Y = y) \tag{2.12}$$

This can be easily extended to three or more dimensions. Moreover, we might be interested in computing some statistics of X, such as the average value assumed by X.

Definition 2.2.8 (**Expected Value**)**.** The expected value $\mathbb{E}$ of a random variable X is defined as:

$$\mathbb{E}(X) = \sum\nolimits_x xP(X = x) \qquad \text{(discrete case)} \tag{2.13}$$

$$\mathbb{E}(X) = \int_{\mathbb{R}} xf_X(x) \qquad \text{(continuous case)} \tag{2.14}$$

where we assume, for simplicity, that the sum (or integral) is convergent.

2.2.2 Useful Distributions

Binomial Distribution. The binomial distribution is a discrete probability distribution that models the number of k successes in n trials for a variable with p the probability of success and $q = 1 - p$ the probability of failure. Its p.m.f is

$$P(x|n, p) = \binom{n}{k} p^k q^{(n-k)}. \tag{2.15}$$

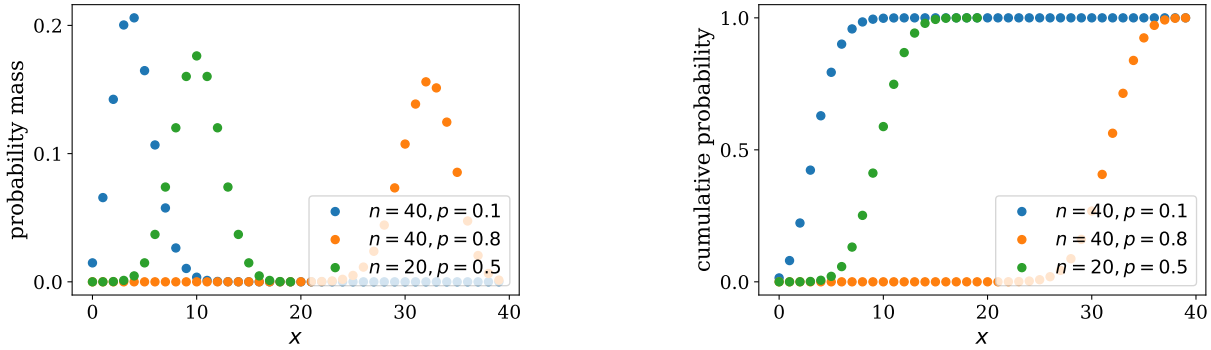


Figure 2.4: Example of a Gamma distribution with different scale and rate parameters.

Multinomial Distribution. The multinomial distribution is a generalization of the binomial distribution. Given k categories, it models the number of successes x_k of each category in n trials, where each of the k category has a fixed success probability p_k. The p.m.f. is:

$$P(x_1, \dots, x_k | n, p_1, \dots p_k) = \frac{n!}{x_1! \cdots x_k!} p_1^{(x_1)} \cdots p_k^{(x_k)}, \qquad \text{with } \sum\nolimits_k x_k = n. \tag{2.16}$$

Categorical Distribution. The categorical distribution represents a special case of the multinomial distribution where $k > 2$ and $n = 1$. Consequently, the p.m.f. can be simply obtained as:

$$P(X = i | p_1, \dots p_k) = p_i. \tag{2.17}$$

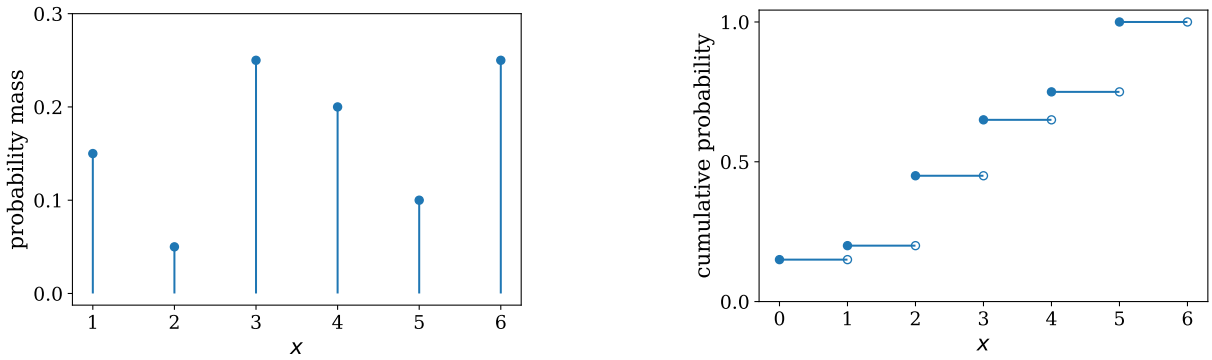


Figure 2.5: Example of a Categorical distribution with six possible states.

Gamma Distribution. The Gamma distribution $Gamma(\alpha, \beta)$ is a continuous probability distribution with the following p.d.f.:

$$P(x|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} e^{-\beta x} x^{\alpha-1}, \qquad x > 0, \tag{2.18}$$

with α the shape parameter, β the rate parameter, and $\Gamma(\alpha) = (\alpha - 1)!$.

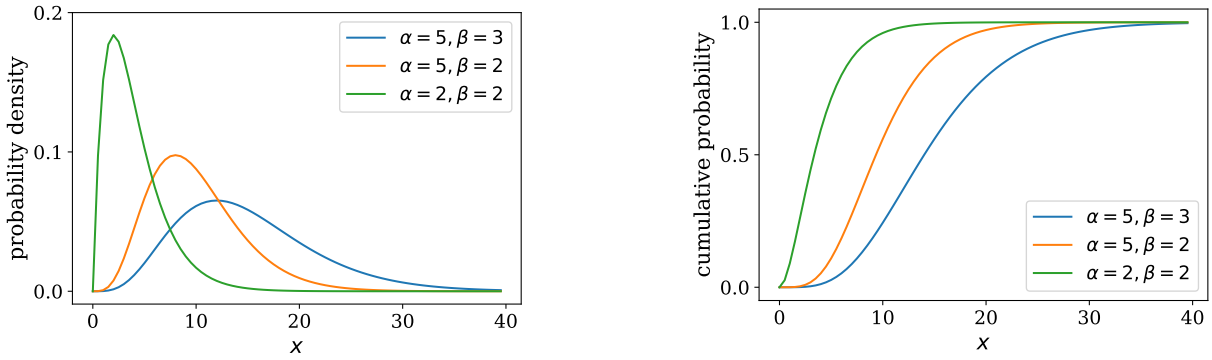


Figure 2.6: Example of a Gamma distribution with different scale and rate parameters.

Gaussian Distribution. The Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ is a continuous probability distribution defined as:

$$P(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\left(\frac{x-\mu}{\sigma^2}\right)^2} \tag{2.19}$$

where $\mu \in \mathbb{R}$ is the mean value and $\sigma^2 > 0$ represents the variance.

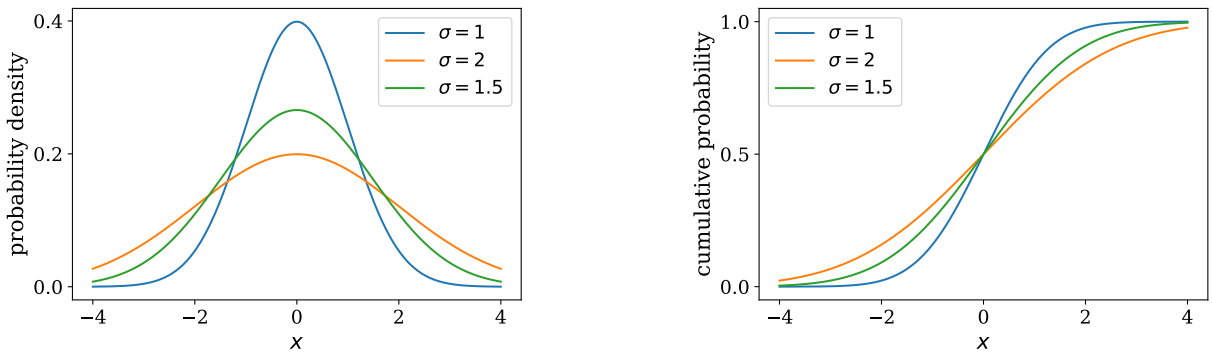


Figure 2.7: Example of a Gaussian distribution with $\mu = 0$ and different values of σ.

2.2.3 Statistical Inference

There are some analogies between statistical inference and machine learning. In particular, the main objective of inference learning is to learn the parameters of the distribution we assume have generated the available data. Similarly, in a machine learning scenario, we want to find the parameters of the model that optimize an objective function. Now, we will introduce further concepts and methodologies that are used throughout this thesis.

Maximum Likelihood Estimation. One of the most popular techniques to find an estimate of the parameters of a distribution is the *maximum likelihood method*. Assuming the samples are independent and identical distributed (i.i.d.), the *likelihood function* is:

$$L(\boldsymbol{\theta}|\mathbf{x}) = L(\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_k|\boldsymbol{x}_1, \dots, \boldsymbol{x}_k) = \prod_{i=i}^{n} f(\boldsymbol{x}_i|\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_k) \tag{2.20}$$

The goal of maximum likelihood estimation (MLE) is to find a value $\hat{\boldsymbol{\theta}}$ that maximizes the likelihood function. Therefore, we aim to find:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}\in\Theta} L(\boldsymbol{\theta}|\mathbf{x}) \tag{2.21}$$

where Θ represents the parameter space. However, given the factorial nature of $L(\boldsymbol{\theta}|\mathbf{x})$, for convenience, we usually work with the so-called log-likelihood $\ell(\boldsymbol{\theta}|\mathbf{x}) = \log L(\boldsymbol{\theta}|\mathbf{x})$. We will make use of the (negative) log-likelihood in Chapters 3 and 4, where it will act as a loss function for learning the textual information available.

Expectation-Maximization Algorithm. An alternative to maximum likelihood estimation is the so-called Expectation-Maximization (EM) algorithm [36]. It iterates the Expectation (E) and Maximization (M) steps until convergence and enables maximum likelihood estimation in latent variable models. More formally, the EM algorithm considers the

'complete-data' likelihood $L(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{Z})$ as an optimization objective to find a solution for the 'incomplete-data' likelihood $L(\boldsymbol{\theta}|\boldsymbol{X})$. In the E-step, the algorithm evaluates the current estimates of the model parameters by computing the expected values of $L(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{Z})$. In the M-step, the algorithm maximizes the expectation computed in the E-step by re-estimating the model parameters. The iterative procedure is guaranteed to converge to the MLE. Starting from some initialized values of the parameters $\boldsymbol{\theta}^{(0)}$, the iterative process consists of the following steps:

1. Compute the expected value $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$ of the log-likelihood $\ell(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{Z})$ using the current estimates of $\boldsymbol{\theta}^{(t)}$:

$$Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) = \mathbb{E}_{\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta}^{(t)}}[\ell(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{Z})] \tag{2.22}$$

2. Find the re-estimated parameters $\boldsymbol{\theta}^{(t+1)}$ that maximize $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$:

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) \tag{2.23}$$

3. Iterate steps 1 and 2 until convergence.

However, as we will see in Chapter 4, the information about the latent variable $\boldsymbol{Z}$ might be unknown. In this case, during the E-step, the algorithm evaluates the current estimates of the model parameters by computing the expected values of the latent variables $\boldsymbol{Z}$. Thus, although the principle remains unvaried, the iterative process is slightly different than before.

2.3 Explainable AI

The adoption of complicated machine learning methods (e.g., deep neural networks) in real-world applications has recently led to an increasing interest in Explainable AI (XAI) [37].

In fact, the improper use of accurate-but-opaque machine learning models may lead to high-impact risks since humans could not fully understand (and, consequently, evaluate) the underlying system or the resulting output. Thus, practical employment of machine learning methods in industrial, medical, and socio-economical applications requires a better understanding of these opaque models through XAI. In this framework, the explanations represent the tool by which we can close the gap between humans and automatic decision models [38]. To introduce the main definitions, challenges, and general taxonomy of XAI, we borrow concepts taken from books, surveys, and taxonomic papers on XAI [8], [39]–[42].

Definition 2.3.1 (**Intrinsic/Post-hoc Explainability**)**.** We refer to intrinsic interpretability when: a) we employ models which are interpretable for definition. This includes, for instance, the use of linear models, decision trees, and rule-based approaches; b) we are able to reduce the complexity of the model during training. For example, a reduced complexity can be achieved through a model that learns how to select a subset of input features leading to the same output as if we were using the complete input information. Post-hoc interpretability alludes, instead, to the practice of using a simpler, interpretable model (i.e., a surrogate model) to explain an already trained black-box model.

Definition 2.3.2 (**Model-specific/Model-agnostic**)**.** A model-agnostic interpretable model is an XAI approach that can be used on different categories of models since it does not need any specific requirement to be applied. Usually, model-agnostic approaches do not require access to the internal mechanisms of the model we want to explain. These models make use of the input features and the corresponding predicted labels to infer the most influential features for the outputs. Contrarily, model-specific approaches are specifically tailored to explain a certain type of predictive model. As such, they need to be adapted accordingly in order to be used for explaining other methods.

Definition 2.3.3 (**Global/Local-level Explanation**)**.** We obtain global interpretability when we can understand and follow the entire logic of the predictive model leading to all

the possible outcomes. An example of global explainability is given by decision trees where, through a flowchart, we can analyze each possible outcome given the considered attributes. Instead, we define local explanations as the ones that allow us to explain only a particular input's prediction at a time. Most of the XAI approaches generate local explanations, which can be further divided into categories depending on the explanation's properties. The explanations that highlight parts of the input are usually referred to as feature-based explanations. Then, the example-based explanations show the most similar training instances to the prediction. Finally, the counterfactual-based explanations attempt to generate an explanation such that we can understand the differences in the input features that would lead to a change in the prediction label.

Properties. According to [39], an explanation should consider the following aspects:

- *Interpretability*: the term refers to which extent a human can understand the provided explanation. As already mentioned, the main challenge is represented by the difficulty in measuring the degree of human-understandability. Given the different nature an explanation can have (textual, visual, gradient-based), it is challenging to have a single metric that could evaluate this aspect. In fact, no common metric exists so far.

- *Accuracy*: it refers to the predictive accuracy of the interpretable model. As explained before, imposing an interpretability constraint during training could affect the flexibility and, in turn, the predictive capabilities of the model.

- *Fidelity*: this term refers to the ability of an interpretable model to mimic the behavior of the black-box one that we want to explain. This property is implicitly obtained when using transparent and faithful models. Contrarily, it is a property we need to take into account when dealing with post-hoc explanation methods.

- *Reliability/Consistency*: the explainable model should be able to maintain certain

levels of precision independently from the model parameter initialization, and high consistency if the explanatory algorithm is applied to the same data instance multiple times.

- *Conciseness*: the explanations, to be easily understandable by the end-users, should be concise. In fact, as reported in [43], sparse explanations would be useful since humans can handle only about 10 cognitive entities at the same time.

An additional property that we should consider, especially in the post-hoc scenario, is the *faithfulness* of the explanations. Intuitively, an explanation is faithful when it can identify the *exact* features that cause a certain prediction. However, there might be some discrepancy between the explanation generated by a post-hoc approach and the actual information used by the model for the same prediction. In fact, even though the post-hoc explanation can match the prediction of the original model, this does not necessarily guarantee that the model uses the same information for its prediction. A discussion about *faithfulness* applied to explanations for graph neural networks can be found in Section 2.4.4.

Explainability Evaluation. As reported in [41], there is no consensus about what explainability is. The reason is that, depending on the task and the data we have, the interpretability objective could be different. As a consequence, a general evaluation framework for explainability does not exist. This, probably, remains one of the main challenges yet to be addressed in XAI. Nevertheless, some efforts have been made during the last few years [44], [45]. For instance, in [45], the authors propose three different approaches for interpretability evaluation based on the type of task (real, simplified, or proxy) and the presence (or not) of human evaluators. To have humans evaluate the quality of the explanations for real tasks looks natural, but the costs in terms of time and money do not always make it feasible in research. On the contrary, generating easier tasks or synthetic data that can be automatically evaluated through some computational metrics may not reflect the complexity of a

real task and, therefore, cannot be effectively used for practical applications. Similar to the faithfulness problem, if we use ground-truth labeled data (i.e., data for which we assume to know the expected explanations), we could end up having some inconsistencies when using post-hoc techniques [46]. In fact, considering the general evaluation pipeline of the explanation methods, there might be a mismatch between the ground truth evidence and the *actual* explanations used by the model. Ground-truth datasets do not consider the weights learned by the model but, instead, represent the explanation that we assume the explainer would generate for a certain input. Consequently, if a post-hoc explanation perfectly matches the ground-truth, this does not assure we are actually explaining the model reasoning. Again, the problem is no longer noteworthy in case we use inherent interpretable models.

Common Post-hoc XAI Techniques. Recent advances in explainable AI propose many post-hoc interpretability approaches which differ depending on the setting and application domain. LIME [47] and SHAP [48] are some of the most popular approaches for local explainability. LIME first generates a new dataset consisting of perturbed samples. Then, it uses a surrogate interpretable model to understand the changes in the outcome when a variation of the original data point is used. SHAP is based on principles of game theory and uses the Shapley values to compute a contribution score for each input feature. By aggregating the computed Shapley values, the approach can also be used for global interpretability. Many extensions of these works have been presented lately [49]–[51]. Attribution methods, instead, try to explain the output by highlighting characteristics of the output itself (e.g., textual explanations with highlighted relevant words) or by highlighting characteristics of the input that strongly influence the result. Similarly to the previous methods, the most popular approaches for feature attribution attempt to understand which portion of the input conditions the output. CAM [52], Integrated Gradients (IG) [53] and GradCAM [54] generate *saliency maps* based on the computed input attribution scores for Convolutional Neural Networks (CNNs). While CAM and GradCAM were specifically tailored for computer vision

tasks, IG has broader applicability. In fact, it only requires the differentiability of the model. In this way, it can compute the gradients of any differentiable model in order to assign the relevant input features with the corresponding output. Following a similar direction, [55] estimates the influence of training examples in neural matrix factorization models through gradient analysis. A comprehensive review of explainability methods for GNNs is presented in Section 2.4.4

An interesting class of interpretable models is represented by logic-based methods. Logic-based approaches create falling rules and interpretable decision sets. In [56], inspired by the need for model explanations in healthcare applications, the authors propose a Bayesian framework to learn falling rule lists which consist of an ordered list of if-then rules determining which training sample should be classified by each rule. [57] proposes to learn decision sets (i.e., independent sets of if-then rules) through an objective function that simultaneously optimizes the accuracy and interpretability of the rule.

Interpretable by Design Models. Other relevant works in the literature face the interpretability problem from different angles. For instance, mechanistic interpretability seeks to understand the inner mechanisms of neural networks by reverse engineering their weights [58]. This approach has been recently used to understand the behaviour of large-language models (LLMs) based on transformers [59]. In Chattopadhyay, Slocum, Haeffele, *et al.* [60], instead, the authors propose to learn predictors by using a composition of interpretable queries to create a model that is interpretable by design. This is similar, in the intent of creating interpretable by desing approaches, to the learning to explain (L2X) paradigm presented in Chen, Song, Wainwright, *et al.* [12]. In this work, the authors propose an instance-wise feature selection method that learns to select a small subset of the input features during training making the model interpretable and the generated explanations faithful by design.

For a comprehensive discussion on methods for explainable AI, we refer the reader to the surveys [39], [40], [61], [62].

2.4 Related Works

In this section, we review the literature related to the methodological chapters that will follow. For each topic, after a general overview of the state-of-the-art, we will discuss limitations and will provide connections with the methods proposed in the next chapters.

2.4.1 Graph Representation Learning

The first attempts to learn low-dimensional representations of graphs were using traditional graph embedding techniques [63]–[66]. After that, more sophisticated methodologies were proposed. Among the approaches for network embedding, the pioneer Deepwalk [67] is undoubtedly one of the most popular. It samples a set of random walks from a graph as sentences and learns node embedding with the SkipGram method [5]. Deepwalk is then extended by LINE [68] to address large-scale networks, and by node2vec [69] to model flexible network neighborhood of nodes. Variations of these works have been recently presented [70]–[72]. Another line of GNN research is TransE [73] and its variants [74], [75]. They generally model graph instances as functions of embedding vectors. Differently, following the message-passing paradigm, graph convolutional neural network and its extensions [30], [76], [77] directly learn a single embedding function which is used to generate all the latent vector representations. However, none of the aforementioned methodologies takes into account any source of human-understandable information such as, for instance, textual data. To include textual data into the embedding, to name a few, [78] learns embeddings of both textual and network structure, and concatenates them to obtain a single embedding for each node. [79]

proposes an extension of DeepWalk that includes the textual information associated with the vertices, and [80] uses a word alignment mechanism to absorb impacts from proximate texts more effectively. In these cases, the textual information is used to improve the quality of the resulting node representations, but it is not exploited to generate explanations of the node embeddings. In this direction, there are few works that aim to improve the interpretability of latent representations. Existing works mainly endeavour to explain the embedding dimensions as clusters in an implicit manner, e.g., employing Canonical Polyadic decomposition [81], or assigning a learned cluster to each vector dimension [82]. Unlike these approaches, the method proposed in Chapter 3 focuses on improving the interpretability of node embeddings in an explicit way. The extra-textual information associated with the graphs is employed to generate word-based vector explanations. In this way, our method maps the latent space of node embeddings into a textual space through such word-based vectors. Consequently, the additional available textual information works as a human-understandable source to generate explanations of node embeddings.

2.4.2 Topographic Organization in Latent Models

Kohonen's seminal work on self-organizing maps (SOM) [83] was introduced in the 1980s. Given the ability to produce low-dimensional representations of high-dimensional data while providing a good approximation of the input space, many extensions and advancements have been proposed in later years. Generative topographic mapping (GTM) [84] is one of the most popular probabilistic alternatives to SOM. GTM provides a generative extension of the SOM, assuming a specific discretized non-Gaussian latent prior. Applications and extensions of SOM span many different domains. For example, in [85], the author proposes a data-driven, statistical approach to visualize large collections of text documents using two-dimensional maps. In collaborative filtering (CF) applications, [86] introduce a topographic

organization of latent classes for rating preferences. Differently from their work, where the user preferences are organized using the numerical information, in Chapter 4, we propose to induce a topographic organization of both user and product latent classes exploiting the associated textual information. As a result, we obtain two separate grids (one for users and one for products) reflecting the word patterns of the data. As reported in [85], the most nuanced and sophisticated medium to express our feelings is our language. Hence, for rating prediction tasks, we believe that it is important to understand and organize the review information in a structured and intuitive way.

2.4.3 Text-based Recommendation Models

Despite the statistical foundation and the nice visualization capabilities of SOMs, with the advent of more powerful and capable machines, researchers started focusing more and more on developing deep neural architectures for recommendation. More generally, for this task, one of the most popular approaches is matrix factorization (MF) and, specifically, Singular Value Decomposition (SVD). This method maps users and items into a latent factor space and computes the rating as a dot product between the user and the product embeddings. Although such approaches are effective and simple, the results are poorly interpretable. Indeed, the embeddings of users and items are not explainable and, not knowing the meaning of each feature, it is impossible to unveil user preferences and product characteristics [87]. Given the availability of large collections of product reviews, researchers have recently extended latent factor models to leverage textual information for improving rating prediction performances. In fact, recent studies in this area tend to conclude that the numerical rating information is not powerful enough for discovering user preferences. One of the first attempts demonstrating the usefulness of leveraging features extracted from reviews to improve the rating prediction accuracy was presented in [88]. Among other popular works in

this direction, Hidden Factors as Topics (HFT) [89] learns topics using a Latent Dirichlet Allocation (LDA)-like model for each item and an MF for ratings. Ratings Meet Reviews (RMR) [90] uses the same LDA model for modeling the textual information, but it uses Gaussian mixtures for the rating prediction part. Similarly, TopicMF [91] learns topics from each review. In [92], instead of using an approach based on LDA, the explored methods are neural network-based. Most of the existing works combine two learning objectives; one (unsupervised) for the textual information, and one (supervised) for rating prediction. The unsupervised loss acts as a regularization term for the rating prediction loss while taking advantage of the review data. Consequently, the vector representations of the reviews are learned to work well for rating prediction tasks while preventing overfitting.

Another category of models uses deep learning approaches for learning latent representations of users and items. These methods mainly differ in the neural architecture they use. In [87], the authors propose an attention-based CNN (Attn+CNN) to build vector representations of users and products. In [56] they propose a hierarchical Bayesian model called Collaborative Deep Learning (CDL), which jointly performs deep representation learning and CF for the rating matrix. In [93], the model learns vector representations of users, items, and reviews, where a review embedding is learned as a translation in the vector space between the user and the product embeddings. In all the above cases, the resulting embedding vectors are only meaningful when compared to each other. For example, considering [93], we can get valuable information about users and products only when the review embeddings are employed. Otherwise, it would be difficult to get interpretable information about them. Differently, in Chapters 3 and 4, the resulting representations are *self-interpretable* providing a 'translation' of the numerical information to human-understandable concepts.

Even though many works have been published in this direction, as pointed out in [94] and [95], it is debatable whether deep-learning-based models are really making progress in this research area. Additionally, despite the majority of the existing works deal with embeddings,

the resulting vector representations usually do not reflect any visualization-driven assumption of the data, making the output poorly explainable. Also, since the ultimate goal is to improve the rating prediction part, the textual information is solely used as an additional source to achieve this goal. Consequently, the latent information contained in the textual data is not fully exploited and investigated.

Moreover, there have been related methods in the direction of explainable recommendations. These works face the problem of generating explanations by using knowledge graph reasoning [96], neural attentive models [97]–[99], attraction modeling [100] or substitute recommendation systems [101]. However, they do not aim to explicitly generate interpretable vector representations. Instead, the main objective is to generate user-specific explainable recommendations using the complete textual information and highlighting words that are important to explain item-specific ratings. Differently, in our work in Chapter 4, we propose a framework for a deeper understanding of the data, presenting a two-step approach that starts from the interpretable organization of the textual information to arrive at the rating prediction task.

2.4.4 Explainability Methods for GNNs

There are several methods to explain the behavior of GNNs. Following Yuan, Yu, Gui, *et al.* [102], explanatory methods for GNNs can be divided into several categories.

Gradient-based methods. [103]–[105]. The main idea is to compute the gradients of the target prediction with respect to the corresponding input data. The larger the gradient values, the higher the importance of the input features.

Perturbation-based methods. [51], [106]–[111]. Here the objective is to study the models' output behavior under input perturbations. When the input is perturbed and we obtain an

output comparable to the original one, we can conclude that the perturbed input information is not important for the current input. Inspired by causal inference methods, [112]–[115] attempt to provide explanations based on factual and counterfactual reasoning.

Surrogate methods. [23], [116]–[119]. First, these approaches generate a local dataset comprised of data points in the neighboring area of the input. The local dataset is assumed to be less complex and, consequently, can be analyzed through a simpler model. Then, a simple and interpretable surrogate model is used to capture local relationships that are used as explanations for the predictions of the original model.

Decomposition methods. [120]–[123]. These methods use decomposition rules to decompose the model predictions leading back to the input space. The prediction is considered as the target score. Then, starting from the output layer, the target score is decomposed at each preceding layer according to the decided decomposition rules. In this way, the initial target score is distributed among the neurons at every layer. Finally, the decomposed terms obtained at the input layer are associated to the input features and used as importance scores of the corresponding nodes and edges.

Model-level methods. [124]. Different from the instance-level methods above, these methods provide a general and high-level understanding of the models. In the context of GNNs, they aim at studying the input patterns that would lead to a certain target prediction. The generated explanations are general and provide a global understanding of the trained GNNs.

Prototype-based methods. [125] propose ProtGNN, a new explanatory method based on prototypes to provide *built-in* explanations, overcoming the limitations of post-hoc techniques. The explanations are obtained following case-based reasoning, where new instances are compared with several learned *prototypes*.

Concept-based methods. [126] propose CGExplainer, a post-hoc explanatory methods for human-in-the-loop concept discovery. This concept representation learning method extracts concept-based explanations that allow the end-user to analyze predictions with a global view. Following a similar idea, [33] propose a concept-based model that generates global and local explanations for graphs through neuron analysis.

Among the methods categorized above, a similar approach in intent is presented in Schlichtkrull, Cao, and Titov [110]. The authors propose a post-hoc technique that learns how to remove the unnecessary edges through layer-wise edge masking. There are two main differences compared to our work: 1) the edge masking is learned from an already trained model, while we learn the edges to remove during training; 2) the edges are treated as independent binary random variables. In our case, instead, the optimization algorithm allows us to model the dependencies between edge variables.

Additional works face the explainability problem from different perspectives as explanation supervision [127], neuron analysis [33], and motif-based generation [128]. For a comprehensive discussion on methods to explain GNNs, we refer the reader to the survey [102]. In Section 5.4, we provide a more detailed comparison with inherent interpretable methods and graph structure learning approaches.

Limitations of Prior Work

When explaining GNNs, we distinguish between how the dataset was constructed and how the GNN makes its predictions. We refer to a *responsible* motif when a dataset is created such that the presence or absence of it determines the class label of the graphs. Hence, the responsible motif represents the underlying evidence (ground truth) allowing us to discriminate among the labels that we hope the explanatory method will find [46]. Instead, when a motif is responsible for the *prediction* of a certain class label, we refer to the edges present

in the motif as the ones *causing* the prediction (*causing* motif). Existing XAI methods for GNNs have several limitations and can lead to inconsistencies. In fact, there could be a mismatch between the responsible motif (ground truth), the actual motif used by the pre-trained model for its prediction (causing motif), and the one identified by the explanatory model (*explanatory* motif) [23], [46]. In contrast, in our work, we know that the prediction of the class label is caused by the explaining motif, as its selection by the upstream model caused the downstream model to make said prediction. As anticipated, we focus on the problem of identifying a subset of the edges as an explanation of the model's message-passing behavior. Hence, an explanation is equivalent to identifying a mask for the adjacency matrix of the original graph. Intuitively, an explanation can be *accurate* and/or *faithful*. It is accurate if it succeeds in identifying the edges in the *input graph* responsible for the graph's class label – i.e., if the explanatory motif matches the responsible motif. This property can, for example, be evaluated with synthetic data where the class label of a graph is determined by the presence or absence of a particular substructure. An explanation is faithful if the edges identified as the explanation *cause* the prediction of the GNN on an input graph – i.e., if the explanatory motif matches the causing motif. Contrary to measuring accuracy, there is no consensus on evaluating faithfulness. Recent work has proposed to measure unfaithfulness as the difference between the predictions of (1) the GNN on a perturbed adjacency matrix and (2) the GNN on the same perturbed adjacency matrix with edges removed by the explanation mask [103], [129], [130]. We believe that this definition is problematic as the perturbation is typically implemented using a swap operation which replaces two existing edges (a, b) and (c, d) with two *new* edges (a, c) and (b, d). Hence, these new edges are present in the unmasked adjacency matrix but not present in the masked one. It is, however, natural that the same GNN would predict highly different label distributions on these two graphs. For instance, consider a chemical compound where we remove and add new bonds. The resulting compounds and their properties can be chemically very different. Hence, contrary to prior work, we define a subgraph to be a faithful explanation, if it is a

significantly smaller subgraph of the input graph and we know that *only its structure* is used in the message-passing operations of equation (2.1).

Chapter Three

Improving the Interpretability of Representation Learning Techniques

3.1 Motivations

Learning graph data with neural networks [30], [67], [69], [73] has recently attracted considerable interest. Graph Neural Networks (GNNs) have been applied to a variety of applications with great success, such as knowledge base completion [75], [131], disease classification and protein interaction prediction [132], [133], machine translation and relation extraction [134]–[137], visual question answering [138], [139], and object detection [140]. In this context, node embedding usually plays an important role. In fact, as seen in the previous chapter, many GNN approaches formulate graph learning with node embedding, and the downstream task (e.g., link prediction, node classification, or (sub-)graph classification) is modeled with node embedding vectors to automatically organize the latent information contained in the data. However, as explained in Section 2.4.1, the resulting embedding vectors are usually not understandable: high-dimensional vectors are difficult to interpret from the human perspective. Thus, for humans to gain trust in AI, along with high performance on graph analysis

tasks, interpretability of node representation is expected. The problem of node embedding interpretability can be faced from two different perspectives: 1. understand the meaning of the generated latent representations; 2. explain why the learning model generates a specific node embedding vector. In this methodological chapter, we intend to explore the first type of interpretability: the meaning of the embedding itself. As textual data are widely available across different domains, we aim to use this source of human-understandable information to interpret embedding vectors. In this way, the generated text-based explanations allow humans to understand the characteristics of the analyzed nodes, the (dis)similarity between different nodes, as well as the results of the follow-up learning tasks, e.g. recommendation. In this context, in addition to the visual evaluation of the results, we introduce two quantitative metrics for text-based explanations with the intent of further helping in their evaluation.

3.2 Model Definition

In this section, we introduce the proposed method, called interpretable graph neural network (iGNN), to learn interpretable node representations. First, we describe how to integrate the textual information within the architecture to learn the probabilistic patterns between words and the considered nodes. Second, we explain how to get user and product-specific word distributions from the results obtained from the training phase.

To illustrate the method, we use a typical review network as a running example. Assume there is a bipartite graph $\mathcal{G}$ with N number of users and M number of products. Between a user i and a product j, there is an edge $e_{i,j}$. Each edge is associated with a multi-set of words (namely, a review of S words) $s_{i,j} = \{w_{i,j,1}, \ldots, w_{i,j,S}\}$, and a rating $r_{i,j}$. The size of the vocabulary is V. The number of reviews is R.

3.2.1 The Generative Process

Technically, the proposed model is based on neural generative modeling. In this way, we are able to integrate the advantages of probabilistic generative models with the ones of neural networks. In particular, the edge sampling and the corresponding probabilistic textual patterns are learned in the following way:

- Each user i is represented by means of an embedding vector $\mathbf{x}_i \in \mathbb{R}^D$ which is automatically learned during training. Taking inspiration from prior works on neural topic modeling [141], [142], we assume the vector is initially drawn from a multivariate Gaussian with zero mean and a diagonal covariance matrix $\mathbf{I}$:

$$\mathbf{x}_i \sim \mathcal{N}_D(\mathbf{0}, \mathbf{I}). \tag{3.1}$$

- For all users, we introduce K clusters. Each user cluster k is associated with an embedding vector $\mathbf{c}_k \in \mathbb{R}^D$, which is again drawn from a Gaussian:

$$\mathbf{c}_k \sim \mathcal{N}_D(\mathbf{0}, \mathbf{I}). \tag{3.2}$$

  Here, we assume all embedding vectors have the same dimension to avoid complicated notation.

- The user cluster weights $\theta_i \in [0, 1]^K$ are computed based on the embedding vectors as:

$$\theta_{i,k} = \text{sparsegen-lin}\left(f(\mathbf{x}_i, \mathbf{c}_k; \phi); \lambda_u\right), \tag{3.3}$$

  which specifies the probability that the user i belongs to cluster k. The function f can be any learnable function (in our case, a fully connected layer) that takes in input the user embedding $\mathbf{x}_i$ and the cluster embedding $\mathbf{c}_k$. In other words, the function f quantifies the relevance or similarity (unnormalized probability) between the user $\mathbf{x}_i$ and the cluster $\mathbf{c}_k$. The output layer is sparsegen-lin, where the hyperparameter λ_u

represents a regularization term shared among all users. Sparsegen-lin is a controllable extension of sparsemax [143], defined in [144] as:

$$\text{sparsegen-lin}(\hat{z};\lambda) = \text{sparsemax}\left(\frac{\hat{z}}{1-\lambda}\right), \tag{3.4}$$

where $\hat{z}$ and $\lambda < 1$ represent the logits and the coefficient to control the regularization strength respectively. In particular for $\lambda \to 1^-$, the probability distribution has the minimum support (i.e. *hardmax*) whereas for $\lambda \to -\infty$, the resulting distribution is non-sparse (i.e. *uniform*).

- Analogously, for each product j, we can proceed in an equivalent manner introducing L clusters and generating $\mathbf{x}_j$, $\mathbf{c}_\ell$ and $\theta_{j,\ell}$ employing a different fully-connected layer g and using the hyperparameters ξ and λ_p accordingly.

- Once we have generated the user and product cluster weights θ_i and θ_j, given a word v, we can extract the corresponding cluster assignment probabilities $\mathbf{z}_{i,v}$ and $\mathbf{z}_{j,v}$. Finally, we can sample a text associated with an edge $e_{i,j}$ by drawing each word $w_{i,j,v}$ in the text as follows:

$$\begin{aligned} \mathbf{z}_{i,v} &\sim \text{Categorical}(\theta_i) \\ \mathbf{z}_{j,v} &\sim \text{Categorical}(\theta_j) \\ \mathbf{w}_{i,j,v} &\sim \text{Categorical}(\beta, \mathbf{z}_{i,v}, \mathbf{z}_{j,v}), \end{aligned} \tag{3.5}$$

where β is a 3D tensor representing the probabilistic patterns among user clusters, product clusters and words. In particular, $\beta_{k,\ell,:}$ specifies a categorical word distribution conditioned on the user cluster k and the product cluster ℓ. It lies in a $(V-1)$-dimensional simplex Δ^{V-1}, such that $\sum_{v=1}^{V} \beta_{k,\ell,v} = 1$ and $\beta_{k,\ell,v} \geq 0$. V denotes the number of words. The parameter $\beta_{k,\ell,v}$ is computed as:

$$\beta_{k,\ell,v} = \text{sparsemax}(\psi(\mathbf{c}_k, \mathbf{c}_\ell, \mathbf{x}_v; \rho)). \tag{3.6}$$

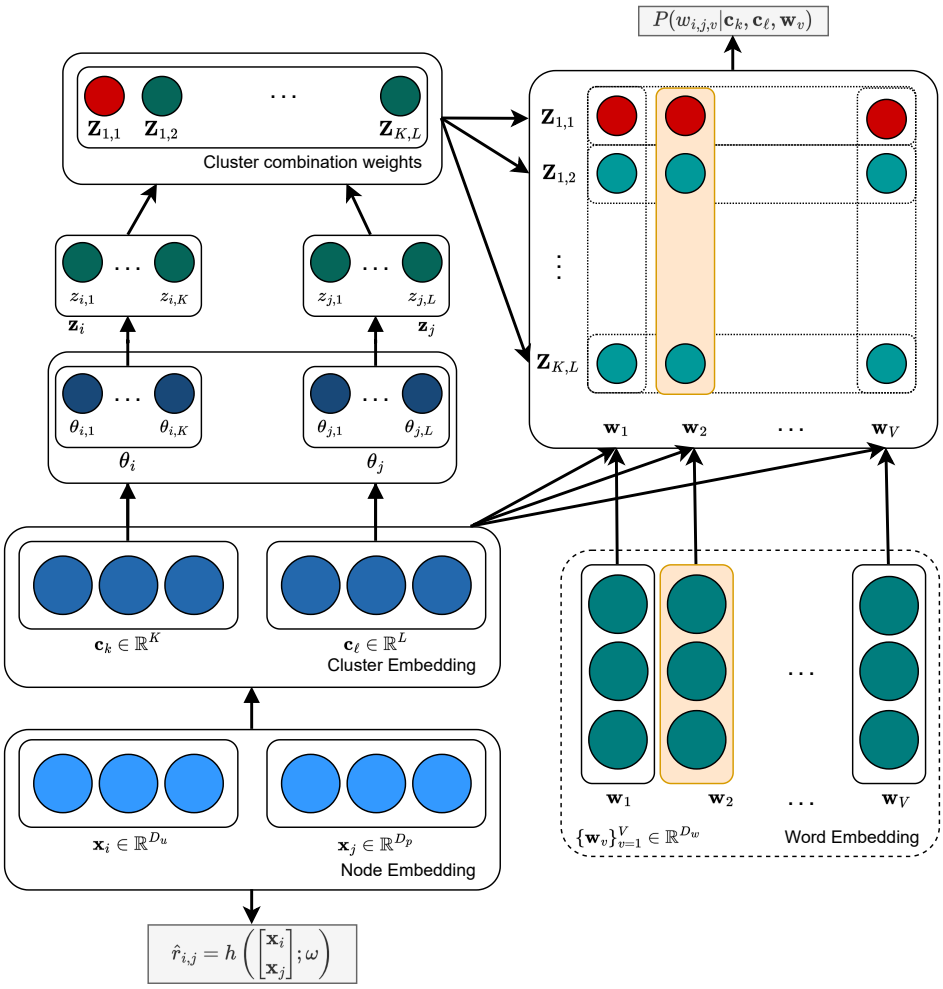


Figure 3.1: The schematic view of the iGNN model. Dashed boxes represent input (non-trainable) data. The line connections depict the dependencies between the variables mentioned in Section 3.2.1.

The function ψ defines again a fully connected layer with parameters ρ. In this case, $\mathbf{x}_v$ represents the pre-trained vector of the word v that has been learned from Word2Vec [5]. Word sampling is inspired by topic modeling. In this case, we assume that all relations and words follow a distribution with distinct parameter values derived from the embedding vectors of the nodes involved. The model is schematically represented in Figure 3.1.

Generation of Textual Explanations of Node Embeddings. Once we learned the cluster assignments $\theta_{i,k}$ and $\theta_{j,\ell}$ for each user i and product j, and the word-based distributions $\beta_{k,\ell,\cdot}$ associated to each cluster combination, it is possible to generate textual explanations of the examined node embeddings. For a node, e.g. a user i, the textual explanation is formulated as a node-specific word distribution $p(\mathbf{w}_v|\mathbf{x}_i)$ conditioned on its embedding vector $\mathbf{x}_i$. In particular, the probability of a word v being used to explain the embedding $\mathbf{x}_i$ is computed as:

$$p(\mathbf{w}_v|\mathbf{x}_i) = \frac{1}{L} \sum_{k,\ell} \theta_{i,k} \ \beta_{k,\ell,v}. \tag{3.7}$$

This is a marginal distribution over all possible user and product clusters. Since the user distribution is not related to any specific products, we marginalize over the product clusters, i.e. the term $1/L$ in (3.7). In a similar way, textual explanations can be generated for product nodes.

3.2.2 Inference and Learning

In summary, our approach attempts to combine two objectives: a) learn traditional node embeddings that perform well in the downstream task of interest; b) generate human-understandable explanations associated to such learned vector representations. Consequently, the learning of the node embeddings and the corresponding textual explanations are driven by two different learning objectives. The first objective is the log-likelihood of the review corpus. For the textual part, the parameters to be learned include embedding vectors of clusters $\{\mathbf{c}_k\}_{k=1}^{K}$ and $\{\mathbf{c}_\ell\}_{\ell=1}^{L}$, and parameters ϕ, ξ, ρ which define the learnable functions f,

g, and ψ respectively. Thus, the log-likelihood of an edge and the corresponding text is

$$\begin{aligned}\mathcal{L}_1 = &\log p(e_{i,j}|\mathbf{x}_i, \mathbf{x}_j, \gamma) + \\ &+ \sum_{v=1}^{S} \log \left(\sum_{k=1}^{K} \sum_{\ell=1}^{L} p(\mathbf{z}_i = k|\mathbf{x}_i, \mathbf{c}_k, \phi) \right. \\ &\left. p(\mathbf{z}_j = \ell|\mathbf{x}_j, \mathbf{c}_\ell, \xi) p(\mathbf{w}_{i,j,v}|\mathbf{c}_k, \mathbf{c}_\ell, \mathbf{x}_v, \rho) \right),\end{aligned} \tag{3.8}$$

where the first term represents the probability that an edge exists between two nodes. This is implicitly included in the computation of the textual information since we suppose that each edge, if exists, has some associated text. The second objective is the error of the predictions. In our study case, the metric used to compute the rating prediction error is the Mean Squared Error (MSE) defined as:

$$\mathcal{L}_2 = \frac{1}{R} \sum (\hat{r}_{i,j} - r_{i,j})^2 \tag{3.9}$$

with

$$\hat{r}_{i,j} = h\left(\begin{bmatrix} \mathbf{x}_i \\ \mathbf{x}_j \end{bmatrix} ; \omega \right), \tag{3.10}$$

where $h(\cdot)$ can be any complex function. In our case, $h(\cdot)$ defines a neural network with the concatenation of the node embeddings $\mathbf{x}_i$ and $\mathbf{x}_j$ as input, and ω as hyperparameters.

Finally, we can define the complete objective function as:

$$\min_{\Theta} \mathcal{L} = \min_{\Theta} (\mathcal{L}_1 + \mu \mathcal{L}_2), \tag{3.11}$$

where Θ represents the parametric space and μ is a hyperparameter to trade-off the importance of the two objective functions. Although the final goal is to generate textual explanations for node embedding, prediction accuracy is crucial during the learning phase. Looking at (3.7), one can see that the textual explanation is conditioned on the node embedding vectors. Hence, we should aim at learning 'meaningful' node embeddings, i.e., obtaining vector representations that are learned to perform well on the given learning task (Eqs. (3.9)

and (3.10)). In this way, by jointly learning embedding vectors, rating predictions and textual explanations, we strengthen the interpretability of the results since the two objectives simultaneously influence each other during the training phase. Finally, given the loss defined in (3.11), we can use backpropagation to efficiently optimize the model.

3.3 Experiments

In the following section, we describe datasets, experimental settings and procedures used to study the effectiveness of the proposed model. First, we want to evaluate the effectiveness of the model on learning node embeddings that perform well in a downstream task, i.e., a rating prediction task. Then, we want to evaluate the generated textual explanations of the node vectors from multiple perspectives, considering both the qualitative and quantitative aspects of their interpretability.

3.3.1 Datasets and Settings.

Datasets. We evaluate our method with a popular benchmark dataset: *Amazon Product Data*[1]. The dataset contains millions of product reviews and metadata, categorized according to the product category, collected between May 1999 and July 2014. We focus on the 5-core version of the datasets, where each user and item has at least five reviews associated with it. A rating, i.e., a labeled link between a user and an item, is an integer value ranging from 1 to 5. To preprocess the texts associated with the review graphs, we embed the words into a 200−dimensional vector space using Word2Vec [5]. We train the word vectors with the raw reviews to capture the semantic and syntactic structure of the corpus. The raw text is preprocessed and cleaned via the following steps: (a) maximum length of a raw review set to

[1] http://jmcauley.ucsd.edu/data/amazon

300; (b) case normalization; (c) removal of stopwords, numbers and special characters; (d) filtering out non-existing words using an English vocabulary; (e) lemmatization; (f) removal of single-word reviews. Once the reviews are normalized, we perform the vocabulary selection independently for each category. This is motivated by the fact that the textual information shows strong diversity depending on the product category. For instance, words frequently occurring in the *Baby* category would not occur in *Office product* reviews. Let $\mathcal{C}$ denote the review collection for a given category, we consider each review $s_{ij} \in \mathcal{C}$ as a document. For each word $w \in s_{ij}$ we compute the corresponding *tf-idf* index and extract the top 10% words (at most 10 for longer reviews) with respect to such index. Then, we merge all the top-words extracted from the corpus $\mathcal{C}$ and we sort them with respect to the *tf-idf* score. The first V words in this ranking denote the vocabulary for the given category. Finally, we further filter the reviews to remove the ones without any word belonging to the vocabulary. To tackle the word co-occurrence sparsity problem over short texts, we extract biterms for each document [145]. For a node-related document with v words, the resulting biterm extraction results in $v(v-1)/2$ unordered word-pairs. In this way, we can pattern the textual information by means of biterms co-occurring in the same document. The statistics of the preprocessed datasets are summarized in Table 3.1. In this chapter, we use data from 12 categories for our evaluation.

Experimental Settings. We set the vector dimensionality $D = 200$ for all users, products and clusters embeddings. We evaluated the robustness of our method to changes in the hyper-parameter D but did not observe any significant performance difference. The number of user clusters $K = 50$ and product clusters $L = 30$; we evaluated the values $[10, 20, 30, 40, 50]$ and we observed that, generally, larger values lead to more sparse cluster assignments. We set the coefficients for the *sparsegen-lin* function as $\lambda_u = 0.9$ and $\lambda_p = 0.75$. The function $h(\cdot)$, defined in (3.10), is a neural network with four hidden fully-connected layers $[128, 64, 64, 32]$. The experiment was run for 200 learning iterations and validated every 2 iterations. A single

Table 3.1: Statistics of the preprocessed datasets.

	Reviews	Users	Items
Amazon Instant Videos	36241	5127	1685
Automotive	20285	2926	1835
Baby	156335	19442	7050
Beauty	194130	22356	12101
Cell Phones and Acc.	191336	27864	10429
Digital Music	64260	5539	3568
Grocery and Gourmet Food	148902	14675	8713
Health and Personal Care	333274	38583	18534
Musical Instruments	10218	1427	900
Office Products	52988	4902	2420
Patio, Lawn and Garden	13213	1684	962
Pet Supplies	152367	19848	8510
Sports and Outdoors	289181	35588	18357
Tools and Home Improv.	133445	16634	10217
Toys and Games	161603	19405	11924
Videogames	227859	24291	10672

epoch performs RMSProp with a learning rate set to 2e-6 and batch size of 256[2].

3.4 Results

We discuss our findings starting from the quantitative analysis of the rating prediction part. After the evaluation of the predictive performance for ratings, we investigate the capacity of our model to generate human-understandable vector representations. In particular, we will analyze both visually and quantitatively the nature of the textual explanations.

Rating Prediction Results. We compare our method with several baselines considering both factorization-based approaches, like SVD and NMF [146], and review-based approaches

[2]The training time for a single batch is around 30 times higher when using the textual information ($\sim$ 1.3s) than the same model without it ($\sim$ 0.04s).

Table 3.2: MSE for iGNN and state-of-the-art approaches.

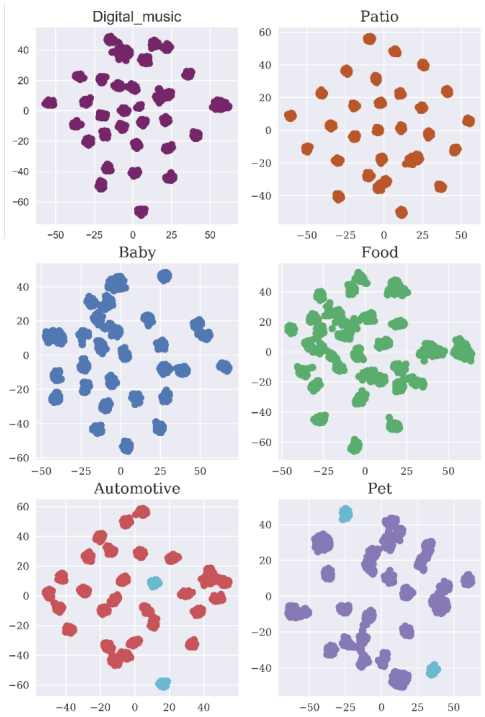
Category	Offset	Attn+CNN	NMF	SVD	HFT	DeepCoNN	TransRev	iGNN
Instant Videos	1.180	0.936	0.946	0.904	0.888	0.943	**0.884**	0.923
Automotive	0.948	0.881	0.876	0.857	0.862	**0.753**	0.855	0.827
Baby	1.262	1.176	1.171	1.108	1.104	1.154	1.100	**1.094**
Beauty	1.322	-	1.204	1.168	1.165	1.184	1.158	**1.073**
Cell Phones	1.451	-	1.357	1.290	1.285	1.365	1.279	**1.161**
Digital Music	1.137	-	0.805	0.797	0.793	0.835	**0.782**	0.855
Gourmet Food	1.165	1.004	0.985	0.964	0.961	0.973	0.957	**0.924**
Office Products	0.876	0.726	0.742	0.727	0.727	0.738	0.724	**0.665**
Patio	1.156	0.999	0.958	0.950	0.956	1.070	**0.941**	**0.941**
Pet Supplies	1.354	1.236	1.241	1.198	1.194	1.281	1.191	**1.186**
Tools and Home	1.017	0.938	0.908	0.884	0.884	0.946	**0.879**	**0.879**
Toys and Games	0.975	-	0.821	0.788	0.784	0.851	0.784	**0.775**

(i.e., methods that exploit textual information for improving the rating prediction performance), including HFT [89], DeepCoNN [147], TransRev [93] and Attn+CNN [87]. We also compare our method with a simple baseline (offset) that uses the average rating score of the training set as prediction.

Following previous works, the data are randomly split by reviews into training (80%), validation (10%) and test (10%) sets. We independently repeat each experiment on five different random splits and report the averaged Mean Squared Error (MSE) to quantitatively evaluate the results. For a fair comparison, the same preprocessed data and data splits are used as input information for all the considered baselines. Table 3.2 summarizes the results of the compared approaches. Note that we primarily focus on the generation of textual explanations for node embeddings. The rating prediction evaluation is a logical consequence of the definition of the generation of textual explanation based on (3.7), and the loss (see (3.11)). From the reported results, iGNN outperforms other models on the majority of the datasets, and achieves comparable results on the remaining ones. This clearly confirms the capacity of the proposed method on learning node representations that perform well on the considered task.

Analysis of the Probabilistic Patterns. We now investigate the capacity of our model on generating meaningful textual explanations for the learned node vector representations. To show the advantages of the generated output compared to common embedding techniques, we analyze the results considering the ability of the text-based explanations to provide human-understandable insights when evaluated both singularly and collectively. First, we evaluate the probabilistic patterns between user clusters, product clusters and words by analyzing the learned word distributions $\beta_{\cdot,\cdot,v}$. As defined in (3.6), β specifies the word distributions for each combination of user and product cluster. For each category, to visualize the learned word-vector distributions on a two-dimensional space, the TSNE method [148] is employed for dimensionality reduction. Figure 3.2 illustrates the cluster organization for different categories. In the two-dimensional embedding space, the clusters are distributed differently and, consequently, well separated. In theory, from direct experience using Amazon, we know that each category can be further divided into sub-categories. Therefore, each cluster identified in Figure 3.2 should represent one sub-category of products. To prove this hypothesis, we select a pair of clusters from two different categories (i.e., *Automotive* and *Pet Supplies*), and we compute the corresponding average word distributions. In this way, by looking at the most probable words of the averaged cluster word distributions, one should be able to infer the main *topic* of the selected clusters. The right table in Figure 3.2 reports the most probable words for each of the selected clusters (highlighted in cyan in the left plot). The results validate our hypothesis since, for each cluster, we can infer the sub-category of interest. As an example, the first cluster in the *Pet Supplies* dataset refers to the *grooming* sub-category, while the other one is about *aquariums*. Thus, we demonstrate that the word distributions $\beta_{\cdot,\cdot,v}$ can capture the latent organization of the data and help to find the textual patterns between user and product clusters, enhancing the interpretability of the results. Indeed, knowing the user and product cluster assignments one can find the *sub-categories* highly correlated with the given items.

Evaluation of Generated Textual Explanations. In the previous paragraph, we demonstrated that the learned text-based vectors, similarly to common embedding techniques, can be analyzed collectively. However, differently from typical neural-based representations, our textual vector representations can also be analyzed singularly since they can be considered *self-interpretable*. In this part, to prove that our model is able to generate textual explanations as node-specific word distributions, we attempt to provide both quantitative and a visual evaluation of the node-related explanations. As reported in [44], lacking common metrics for interpretability quality is problematic to the research community to make progress in this area and, in an unsupervised setting, the problem is even more clear. Indeed, it is common for researchers to simply rely on the human visualization of the results, e.g., by employing attention mechanisms or heat maps, in order to make the results human-explainable [37]. For the quantitative analysis of the results, despite the fact that we can still evaluate the quality of the results based just on human perception of the highlighted relevant words,



Automotive		Pet Supplies	
Cluster 1	Cluster 2	Cluster 1	Cluster 2
battery	trailer	work	filter
charge	power	hair	fish
charger	gas	brush	water
power	away	look	gallon
plug	compressor	thing	plant
code	large	fur	heater
cord	pressure	smell	turtle
lead	jeep	quality	flow
transmission	price	wet	clean
phone	guy	stuff	pump
volt	hole	comb	algae
change	weight	shed	shrimp
adapter	fast	groom	work
smell	fuel	flea	gravel
connect	space	shampoo	tube
electronics	*performances*	*grooming*	*aquariums*

Figure 3.2: Cluster organization of the word-vector distributions $\beta_{\cdot,\cdot,v}$ for different categories. The clusters analyzed in the right table are highlighted in cyan.

we attempt to introduce some metrics that could further help in assessing the quality of the results. For the visualization and evaluation of the correlation between the generated word distributions and the node-related words in the data, we adopt the following procedure. Given a sampled node, we first extract the *top*-15 words of the generated word distribution, i.e., the 15 words having the highest probability in the distribution. Second, we extract the set of words associated with the sampled node in the original data. Let denote with A and B respectively, these two sets. The Jaccard similarity is used to measure similarity between two sets of words A and B, which is defined as follows:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}. \tag{3.12}$$

The Jaccard score only considers the direct matches among the words in the two sets considered. However, two words may be different but have a similar meaning. Hence, to consider the semantic similarity of the two sets, we employ the so-called Word Mover's Distance (WDM), introduced by [149]; this metric takes into account the word similarities in the latent space and computes the minimum distance needed by the words in text A to *travel* in the semantic space to reach the words in text B. Since Jaccard and WDM have different scales and behaviors, we apply MinMaxScaler to Jaccard, and transform WDM as follows:

$$\text{t-}WDM(A, B) = 1 - \left(\frac{d_i - min(\mathbf{d})}{max(\mathbf{d}) - min(\mathbf{d})} \right), \tag{3.13}$$

where d_i is the distance between the sets A and B for a node i, and $\mathbf{d}$ represents all the distances between the two sets for each node in the graph. In this way, the transformed distance score is in the range [0,1] and, opposite to the definition of semantic distance, the higher the value the closer are sets A and B.

To analyze the quality of the generated textual explanations, we compute the two scores for all the selected datasets. Figure 3.3 illustrates the value distribution of these metrics across the nodes in the datasets. The Jaccard values mostly range between $[0.4 - 0.7]$, while the WDM scores are highly concentrated in the range $[0.6 - 0.8]$. It is important to note that the

lower Jaccard scores do not impact the performance of the model. Instead, they confirm that the model generates textual explanations that are not redundant, as would be the case with too high similarity scores. Indeed, as written in [37], two sets of words can be semantically similar even with low lexical overlapping. Furthermore, this shows that our approach is not simply based on lexical similarity, but also takes into account the semantic similarity of the words in order to generate textual explanations that differ from the original data and, consequently, provide new insights regarding the items under examination. Figure 3.4 shows an example of the generated word distribution of a sampled node with the corresponding explanatory scores.

We continue the evaluation by analyzing the results on the *Baby*, *Gourmet Food*, and *Pet Supplies* datasets. The example nodes showed in Table 3.3 demonstrate the ability of iGNN to capture the most probable words associated with a given node. In particular, not only direct correspondences with the words in the data but also highly related words. For example, looking at the sampled node from the *Baby* category, we can observe that the generated explanation can be used to capture additional characteristics of the node. In general, it seems that the node is related to toys and strollers. In addition to the information about the types of products of interest, by analyzing the bold blue words, we can infer other characteristics the product has (if the node is a product) or should have (in case the node is a user). In particular, we can deduce that the related items should be safe and made with suitable quality materials. Additionally, we can notice the impact of the scores on the quality

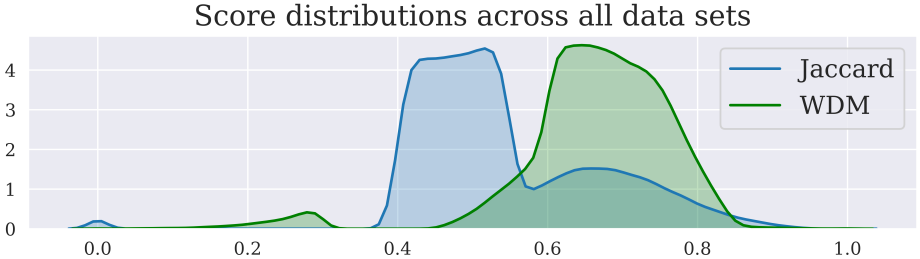
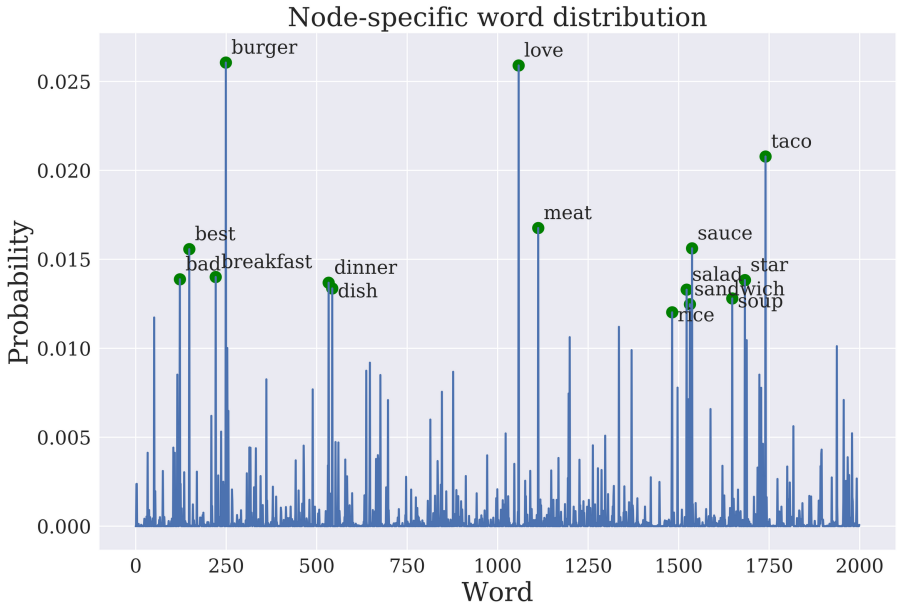


Figure 3.3: Evaluation of the metric distributions across all datasets.

of the generated word distributions. The first two examples, taken from the *Gourmet Food* data set, have different Jaccard scores but similar distances. Nevertheless, the first example looks semantically good. The results show that exploiting word vector representations during the training phase is crucial in order to *push* the probabilities of terms closer together on a semantic level.



Top-15 Words	bad **best** **breakfast** **burger** dinner dish **love** **meat** **rice** **salad** **sandwich** **sauce** soup star **taco**
Node-related Words	ambiance authentic **best** **breakfast** choice cilantro decision guacamole **love** **meat** pas plain satisfied **sauce** spacious staple **taco**
	t-J(A,B) = 0.696 t-WDM(A,B) = 0.718

Figure 3.4: Interpretability case study on a random node. The figure depicts the node-specific word distribution; the 15 highest probabilities are highlighted by green points. Top-15 Words and Node-related Words refer to the sets A and B defined in (3.12)-(3.13). Black bold represents the overlapping words; blue bold highlights words that may explain further characteristics of the node.

We can also observe that (a) the model does not take into account the polarity of the words. Hence, it may be possible to extend and improve the architecture by exploiting sentiment (or rating) information in order to obtain the distribution of *positive*/*negative* words; and (b) the vocabulary selection can be improved by employing more sophisticated methods that may ultimately result in improved performance. In terms of human readability, given the availability of a large number of textual reviews, a more intuitive approach would be to exploit natural language explanations. In fact, a textual explanation would probably be more intelligible than our word-based vector distribution. However, the overhead in terms of data needed and model complexity makes this choice prohibitive in low-resource scenarios. As mentioned in Section 2.4.3, text-based explanations (or recommendations) need the *complete* textual information to generate meaningful natural language outputs. In our case, instead, only a subset of meaningful words for each product category is used. Furthermore, the resulting word-based vector representations can be potentially employed as input features of users and products in recommender system tasks. In a similar intent, we demonstrate it is possible to use user and product latent vectors for rating prediction in the next methodological chapter.

Real-world applications. The idea of extracting human-interpretable information starting from a text-labeled graph can have real-world implications. In the US patent [150] related to the proposed methodology, we present a system for extracting interpretable entity profiles for system management and operations. Complicated systems (e.g., smart hospitals) usually include a variety of entities (e.g., patients, doctors, and medical staff) and rich information (e.g., clinical narratives, medical images of patients, and metadata of patients and doctors). Given the limited amount of human and physical resources (e.g., treatment rooms, specialized doctors), system operators are required to gain a comprehensive overview of all the entities such that they can better manage and operate the system for optimizing different management tasks (e.g., resource allocation). Typically, system operators conduct

entity profiling by collecting all information about a particular entity and then extracting a digital representation of that entity from the collected data. However, entities are generally not independent of each other. Thus, a graph representation of the data can help organize all the relevant entities and their relationships in a structured way. In this framework, given the complexity of the data, AI-based technologies would be beneficial to help operators manage such amounts of data. However, as explained before, general NNs only learn real numbers as digital representations of unstructured data (e.g., images, texts, etc.) that, unfortunately, are not understandable by humans. For this reason, we propose to use the presented work as a way for assisting system operators in their decisions. In this way, through human-understandable word-based profiles, the operators can a) have an interpretable way to operate and explain the system; b) use the resulting word-based profiles to explore and discover similarities among different entities in the system, facilitating and accelerating different management tasks. The proposed idea can be extended or adapted and, consequently, is not limited to the example of smart hospitals.

3.5 Summary

In this chapter, we present interpretable graph neural networks (iGNNs), which learn human-understandable explanations of node embeddings based on extra-textual information associated with graphs. In order to enhance the interpretability of the results, during training the model simultaneously learns node embeddings and verbal descriptions. As a result of the introduction of node cluster embeddings, the model can discover patterns among nodes, ratings, and textual information. In this way, the proposed methodology acquires the discrete structure of graph data and text-based explanations in an elegant manner. Finally, recalling Eqs. (3.9) and (3.10), our model offers a flexible framework that can be integrated with different learning tasks. Due to the availability of rating information, we perform a

rating prediction task. However, the second term of the complete objective function defined in (3.11) can be modified depending on the intended application. According to the results of the explanation analysis, the model is capable of generating human-understandable explanations while performing competitively on a prediction task. Indeed, the model is able to explain the meaning of the learned node representations in a human-interpretable way. A quantitative analysis of the textual explanations adds value to the assessment of the quality of the results. Unlike the simple visualization of sampled outputs, the proposed investigation provides a comprehensive understanding of the general model's behavior based on the textual explanation. The approach may be extended in future work by integrating the polarity of the words, and using the interpretation of the results for downstream purposes, such as human-understandable recommendations.

Table 3.3: Evaluation of retrieved nodes from different datasets. Black bold represents the matching words; blue bold highlights words that may explain further characteristics of the node.

Gourmet Food	
Top-15 Words	**butter buy cereal chocolate** cracker **delicious** eat **energy honey milk nut package** price **say time**
Node-related Words	almond alternative arrive **butter buy** container creamy date deal **delicious** healthy jar mess **nut** oil oily order **package** peanut plastic pull **say** size stir **time** variety way
	t-J(A,B) = 0.539 t-WDM(A,B) = 0.748
Top-15 Words	**best** buy **chocolate cup dark delicious** eat mix nice **organic price say strong tea time**
Node-related Words	agree arrive bad **best** big **chocolate** cookie crack **cup delicious** disappear early equal expect forever heat package picky **price** quite **say** shelf sleeve staff **tea time** week
	t-J(A,B) = 0.640 t-WDM(A,B) = 0.753
Baby	
Top-15 Words	**color** cup cute **daughter** hard **item material nice perfect product** pump **quality safe stroller toy**
Node-related Words	bouncer **color** comfy **daughter** flop happy insert issue **item** lot **perfect product** return **stroller** swing tiny **toy**
	t-J(A,B) = 0.777 t-WDM(A,B) = 0.731
Pet Supplies	
Top-15 Words	**clean color** eat **filter good** help **long look** order **size small thing water week work**
Node-related Words	bottle brush care chip **clean** daily drink excite **good** last **long look** nice puppy **small** smell spray stay **thing** triple type **water week** white **work**
	t-J(A,B) = 0.943 t-WDM(A,B) = 0.893

Chapter Four

An Interpretable Alternative to Neural Representation Learning

4.1 Motivations

In recent years, as previously mentioned in Chapter 1, with the advent of more powerful and capable machines, researchers have started focusing more and more on developing (deep) neural architectures for a wide range of applications. The success of neural-based approaches in different domains, as language modeling [151], [152] or computer vision [153], [154], led these models to also dominate the recommender systems research area [77], [155], [156]. Nonetheless, it is not necessarily true that models with high complexity are inherently more accurate [8], [40]. In fact, for recommendation tasks, recent works have raised some concerns about the relative performance improvements of deep learning approaches compared to simpler algorithms [94], [95], [157]. Additional works [9], [158] showed that new recommendation methods do not significantly outperform existing approaches or even can be outperformed by very simple methods, e.g. nearest-neighbor-based techniques [159]. Previous investigations in this direction were mainly focused on pure collaborative filter-

ing (CF) data, where the only available input was the rating matrix. Nevertheless, recent studies in this area tend to conclude that numerical rating data are not informative enough for discovering user preferences. Consequently, given the availability of large collections of textual data, such as product reviews or social media posts, many approaches have tried to extend and improve recommendation models by leveraging such textual information [78], [89], [93], [147], [160]. In fact, corpora of textual documents contain a wealth of information. Textual corpora can be used to improve the predictive performances of recommendation systems, but more importantly, they provide human understandable explanation about user preference and product properties. As mentioned in the previous chapters, to integrate the extra textual information into recommendation systems, most existing works employ embedding methods, i.e., representing items, words, and documents as high-dimensional numerical vectors for better flexibility. Although the embedding-based methods often provide good predictions, as seen before, the resulting latent representations are usually not explainable; if singularly evaluated, a 100D or 200D vectors can be hard to comprehend by humans.

To complete the study on the interpretability of the embeddings started in the previous chapter, inspired by recent discussions about the *"neural hype"* [9], in this methodological chapter we investigate whether using a simpler, more transparent and principled way to learn user and product latent representations can lead to comparable results in the recommendation task, i.e. review-based rating prediction. Differently from Chapter 3, we present a probabilistic framework for the topographic organization of review data where the resulting latent codes are rather compact and sparse. In contrast with previous neural-based works, we impose a double two-dimensional topological organization of user and product latent classes based on the textual information. As a result, the latent classes of users and products are organized on two different square grids that reflect the textual input space. The grid organization makes the investigation and interpretation of the results fast and intuitive. Additionally, the probabilistic assumptions of our system enable us to analyze the

extracted information in a statistical manner. With the interpretable topographically organized latent space representations obtained in our probabilistic framework, one can naturally solve many downstream learning tasks, such as product rating prediction. Motivated by the strong correlation between reviews and ratings, we believe that exploiting our simple, yet meaningful representations of textual review information will lead to competitive rating prediction results, while maintaining the explainability of the latent classes. We compare our results with state-of-the-art approaches, including both neural network (NN) based methods and non-NN ones. The main goal of this work is not to develop a new approach that outperforms the state-of-the-art with respect to a single predictive score (e.g. accuracy or area-under-curve (AUC)), but rather to propose a principled and interpretable method that focuses on learning a transparent latent structure of the review data, while having competitive rating prediction performances. Finally, inspired by the recent debates on the *phantom progress* [157] of NN-based methods for recommendation tasks, we compare our findings in terms of both interpretability and predictive performances with respect to the most popular text-based neural approaches for rating prediction.

In summary, the main contributions of this chapter are threefold:

1. We present a topographic organization of user and product latent classes based on the latent structure of the review data. Common embedding techniques model *each* item by employing dense and complicated representations. In our case, we model *classes* of users and products resulting on latent vectors that are compact, interpretable and rather discrete. Also, different from existing works where the analysis of the user and product characteristics from the textual perspective is practically ignored, we propose a more complete investigation of the available information.

2. In contrast with previous works where the interpretability is provided by *post-hoc* evaluations, the probabilistic assumptions of the framework allow us to explicitly impose

through model formulation the interpretability of the latent codes. In addition, the topographic organization can help us to understand the relationships among different latent classes. Indeed, classes close to each other in the topographic maps should exhibit similar patterns.

3. Motivated by the strong correlation between reviews and ratings, we exploit our representations of the review information as input for a rating prediction task. We contribute to the debate about the effective performances of neural-based approaches in comparison with more principled and simpler methods. Differently from previous investigations, we present a comprehensive comparison mainly focused on text-based approaches for rating prediction, considering both interpretability capacity and predictive performances.

4.2 Model Definition

In this section, we present the key ingredients of our framework. Based on our observation that analysis of the textual latent patterns is often not fully covered, we propose an interpretable review-based probabilistic model for rating prediction. First, we describe the estimation of the model parameters. Then, to make our model able to integrate new user data after training, we propose an out-of-sample extension allowing us to compute the latent class assignments of new users that reviewed products available in our data set. Finally, we present how to exploit the estimated latent space representations as model inputs for a rating prediction task.

4.2.1 Topological Organization of the Latent Model

Consider a collection of users $\mathcal{U} = \{u_1, u_2, ..., u_N\}$, products $\mathcal{P} = \{p_1, p_2, ..., p_M\}$ and words $\mathcal{V} = \{w_1, w_2, ..., w_V\}$. The data $\mathcal{D}$ is a collection of R triples $\mathcal{D} = \{(u^i, p^i, r^i)\}_{i=1}^{R}$, each triple identifying the user $u^i \in \mathcal{U}$ writing a review r^i on product $p^i \in \mathcal{P}$. The review r^i is a multi-set of words from $\mathcal{V}$, $r^i = (w_1^i, w_2^i, ..., w_{S_i}^i)$, $w_j^i \in \mathcal{V}$. The latent variables $\mathbf{z}_u \in \{1, \ldots, K\}$ and $\mathbf{z}_p \in \{1, \ldots, L\}$ represent *abstract* classes of users and products.[1]

Given a review i, the probability of sampling a word $w_j \in r^i$ is modeled as:

$$P(w_j^i|u^i, p^i) \;=\; \sum_{k=1}^{K}\sum_{\ell=1}^{L}\Big(P(w_j^i|\mathbf{z}_u = k, \mathbf{z}_p = \ell)P(\mathbf{z}_u = k|u^i)P(\mathbf{z}_p = \ell|p^i)\Big). \tag{4.1}$$

We impose a grid topology on latent classes via the channel noise methodology [85], [86]. Let's assume $\mathbf{y}$ and $\mathbf{z}$ are two different latent classes in our grid. The channel noise for both the product and user latent class grids is defined using the neighborhood function:

$$P(\mathbf{y}|\mathbf{z}) = \frac{\exp\left(\frac{-\|\mathbf{z}-\mathbf{y}\|^2}{2\sigma^2}\right)}{\sum_{\mathbf{y}'}\exp\left(\frac{-\|\mathbf{z}-\mathbf{y}'\|^2}{2\sigma^2}\right)} \tag{4.2}$$

where $\sigma > 0$ controls the 'concentration' of the transition probabilities among the neighbors of the latent class $\mathbf{y}$.[2] Overall, when $\mathbf{y}$ and $\mathbf{z}$ are close to each other on the grid, the probability of being corrupted one into another is higher than when they are distant. Additionally, when σ is close to 0, then the transition probabilities are more concentrated around $\mathbf{y}$ than for larger values of σ. For each user $u \in \mathcal{U}$, the generative process is as follows:

1. the latent class assignment $\mathbf{z}_u = k$ is randomly sampled from the user-conditional probability distribution $P(\cdot|u)$ on $\mathbf{z}_u$;

[1] More formally, z_u is a high-dimensional vector whose index u takes values from 1 to K. To simplify notation, in order to state that z_u points to the k-th point grid we will simply write $z_u = k$.

[2] Since the channel noise formulation is the same for both the user and product latent grids we do not use subscripts u or p when referring to the latent class.

2. the latent class assignment may be corrupted by the channel noise $P(\mathbf{y}_u|\mathbf{z}_u = k)$, resulting in a (possibly new) class assignment $\mathbf{y}_u = k'$.

For each product $p \in \mathcal{P}$, we can proceed equivalently to get possibly corrupted assignments $\mathbf{y}_p = \ell'$ starting from the sampled assignments $\mathbf{z}_p = \ell$. The topographic organization has a twofold advantage: 1) it enables a model-based visualization tool of the word patterns that can be easily investigated; 2) it regularises the latent class model and hence prevents *overfitting*, while employing a large number of latent classes [86]. Finally, the model has now the following form:

$$P(\mathbf{y}_u = k'|u^i) = \sum_{k=1}^{K} P(\mathbf{y}_u = k'|\mathbf{z}_u = k)P(\mathbf{z}_u = k|u^i) \tag{4.3}$$

$$P(\mathbf{y}_p = \ell'|p^i) = \sum_{\ell=1}^{L} P(\mathbf{y}_p = \ell'|\mathbf{z}_p = \ell)P(\mathbf{z}_p = \ell|p^i) \tag{4.4}$$

$$P(w_j^i|u^i, p^i) = \sum_{k'=1}^{K}\sum_{\ell'=1}^{L}\Big(P(w_j^i|\mathbf{y}_u = k', \mathbf{y}_p = \ell')P(\mathbf{y}_u = k'|u^i)P(\mathbf{y}_p = \ell'|p^i)\Big). \tag{4.5}$$

In our model the class-conditional word distribution $P(w_j^i|\mathbf{y}_u = k', \mathbf{y}_p = \ell')$ is formulated as multinomial.

4.2.2 Inference and Learning

Assuming independent data items in $\mathcal{D}$, the log-likelihood of the model is:

$$\mathcal{L} = \sum_{i=1}^{R} \log P(r^i|u^i, p^i). \tag{4.6}$$

We will consider a simple review model assuming independence of the words appearing in the review i:

$$P(r^i|u^i, p^i) = \prod_{j=1}^{S_i} P(w_j^i|u^i, p^i). \tag{4.7}$$

Hence, the log-likelihood reads:

$$\mathcal{L} = \sum_{i=1}^{R} \sum_{j=1}^{S_i} \log P(w_j^i | u^i, p^i). \tag{4.8}$$

Plugging in eqs. (4.3–4.5), we obtain:

$$\begin{aligned} \mathcal{L} = \sum_{i=1}^{R} \sum_{j=1}^{S_i} \log \Bigg[& \sum_{k'=1}^{K} \sum_{\ell'=1}^{L} P(w_j^i | \mathbf{y}_u = k', \mathbf{y}_p = \ell') \\ & \sum_{k=1}^{K} P(\mathbf{y}_u = k' | \mathbf{z}_u = k) P(\mathbf{z}_u = k | u^i) \\ & \sum_{\ell=1}^{L} P(\mathbf{y}_p = \ell' | \mathbf{z}_p = \ell) P(\mathbf{z}_p = \ell | p^i) \Bigg]. \end{aligned} \tag{4.9}$$

For training, we use the Expectation-Maximization (EM) algorithm enabling maximum likelihood estimation (MLE) in latent variable models. It iterates the Expectation (E) and Maximization (M) steps until convergence. Detailed derivations of the following equations are presented in Appendix B.

4.2.3 EM-step

E-step. In the E-step, the algorithm evaluates the current estimates of the model parameters by computing the expected values of the latent variables. We will denote these quantities using $\hat{P}(\cdot|\cdot)$. Note that we have two levels of hidden variables. First, given the user and the product they reviewed, we do not know which latent classes $\mathbf{z}_u$ and $\mathbf{z}_p$ represented the (*user, product*) couple when writing the review. Second, we know that the underlying latent classes $\mathbf{z}_u$ and $\mathbf{z}_p$ may have been disrupted to latent classes $\mathbf{y}_u$ and $\mathbf{y}_p$ before producing the review, but we do not know their identity. To simplify mathematical notation, we will denote $\mathbf{z}_u = k$, $\mathbf{z}_p = \ell$, $\mathbf{y}_u = k'$ and $\mathbf{y}_p = \ell'$ as $\mathbf{z}_u^k$, $\mathbf{z}_p^\ell$, $\mathbf{y}_u^{k'}$ and $\mathbf{y}_p^{\ell'}$, respectively.

$\hat{P}(\mathbf{z}_u^k, \mathbf{z}_p^\ell | u, p, w)$ is evaluated as:

$$\frac{P(\mathbf{z}_u^k | u) P(\mathbf{z}_p^\ell | p) \sum_{k'} \sum_{\ell'} S(k, \ell)}{\sum_{k''} \sum_{\ell''} P(\mathbf{z}_u^k | u) P(\mathbf{z}_p^\ell | p) \sum_{k'} \sum_{\ell'} S(k'', \ell'')} \tag{4.10}$$

with $S(\alpha, \beta) = P(w|\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'})P(\mathbf{y}_u^{k'}|\mathbf{z}_u^{\alpha})P(\mathbf{y}_p^{\ell'}|\mathbf{z}_p^{\beta})$.

Instead, $\hat{P}(\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}|w, u, p)$ is computed as:

$$\frac{P(w|\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}) \sum_k G(k') \sum_\ell W(\ell')}{\sum_{k''} \sum_{\ell''} P(w|\mathbf{y}_u^{k''}, \mathbf{y}_p^{\ell''}) \sum_k G(k'') \sum_\ell W(\ell'')} \tag{4.11}$$

where $G(\alpha) = P(\mathbf{y}_u^{\alpha}|\mathbf{z}_u^{k})P(\mathbf{z}_u^{k}|u)$ and $W(\beta) = P(\mathbf{y}_p^{\beta}|\mathbf{z}_p^{\ell})P(\mathbf{z}_p^{\ell}|p)$.

M-step. In the M-step, the algorithm maximizes the expectation computed in the E-step by re-estimating the model parameters. To do so, we need to specify functional forms of the distributions for $P(\mathbf{z}_u^k|u)$, $P(\mathbf{z}_p^\ell|p)$ and $P(w|\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'})$. It is natural to model these distributions as multinomial distributions. Thus, we assume:

$$P(\mathbf{z}_u^k|u) \sim \text{Multinomial} \tag{4.12}$$

$$P(\mathbf{z}_p^\ell|p) \sim \text{Multinomial} \tag{4.13}$$

$$P(w|\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}) \sim \text{Multinomial.} \tag{4.14}$$

The update equation for $P(w|\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'})$ is:

$$P(w|\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}) = \frac{\sum_{(u,p)\in\mathcal{B}(w)} \hat{P}(\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}|u, p, w)}{\sum_{w'} \sum_{(u,p)\in\mathcal{B}(w')} \hat{P}(\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}|u, p, w')} \tag{4.15}$$

where $\mathcal{B}(w)$ is the set of (*user, product*) tuples associated with the word w.

Denoting the set of words used by user u to review product p by $\mathcal{W}(u, p)$, we obtain the update equations for $P(\mathbf{z}_u^k|u)$ and $P(\mathbf{z}_p^\ell|p)$ as:

$$P(\mathbf{z}_u^k|u) = \frac{\sum_p \sum_{w\in\mathcal{W}(u,p)} \sum_\ell \hat{P}(\mathbf{z}_u^k, \mathbf{z}_p^\ell|u, p, w)}{\sum_p |\mathcal{W}(u, p)|} \tag{4.16}$$

$$P(\mathbf{z}_p^\ell|p) = \frac{\sum_u \sum_{w\in\mathcal{W}(u,p)} \sum_k \hat{P}(\mathbf{z}_u^k, \mathbf{z}_p^\ell|u, p, w)}{\sum_u |\mathcal{W}(u, p)|}, \tag{4.17}$$

where $|\mathcal{W}(u, p)|$ is the size of $\mathcal{W}(u, p)$.

Topographic Initialization with SOM. The successful application of the EM algorithm depends on the initial position in the parameter space since the algorithm can be sensitive to parameter initialization. We follow an approach similar to the one presented in [86]. Different from their work, where the SOM was run on a dataset of user ratings, we run two different instances of SOM (one for users and one for products) using the review information. We set the number of nodes in SOM to be equivalent to the number of latent classes, i.e. K for users and L for products. We denote by $\mathcal{V}$ the vocabulary set. For each user u, there is an associated $|\mathcal{V}|$-dimensional vector $\mathbf{v}_u$. Each vector dimension represents a word w in the vocabulary and the corresponding value is the term frequency, i.e. how many times the user has written a review using that word. The analogous reasoning is valid for products. In the latter case, each vector dimension represents how many times a word w has been used for reviewing the product p.

After training the two instances of SOM, the conditional latent priors for users and products, i.e. $P(\mathbf{z}_u^k|u)$ and $P(\mathbf{z}_p^\ell|p)$ respectively, are computed as follows. If the user u belongs to the SOM cluster node $k \in \mathcal{Z}_u = \{1, \ldots, K\}$, then we *softened* the hard assignment using the following transformation:

$$P(\mathbf{z}_u^k|u) = \begin{cases} A & \text{if } u \in k \\ (1-A)/(K-1) & \text{otherwise.} \end{cases} \tag{4.18}$$

The parameter A is defined such that if the user u belongs to the class k, $P(\mathbf{z}_u^k|u) = A$ should be $B > 1$ times higher than $P(\mathbf{z}_u^{k'}|u)$ for all the other $k' \in \mathcal{Z}_u$. Thus, $A = B/(K-1+B)$. The product conditional latent prior can be generated in an equivalent manner by changing the parameters accordingly.

The empirical distribution for word patterns $P(w|\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'})$ is then computed via the following procedure: (a) to introduce the topology, we follow the steps described in Section 4.2.1 to get (possibly corrupted) class indices $\mathbf{y}_u^{k'}$ and $\mathbf{y}_p^{\ell'}$ for all users u and products p; (b) we denote

with $N(w, k', \ell')$ the number of times the word w has been used by users belonging to the latent class k' to review products that belong to the latent class ℓ'. Thus, the empirical distribution is estimated as:

$$P(w|\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}) = \frac{N(w, k', \ell') + m}{mV + \sum_{w' \in \mathcal{V}} N(w', k', \ell')}. \tag{4.19}$$

Due to sparseness, we apply a Laplace correction for smoothing the empirical distribution. The parameter m is a positive number and, in this case, $m = 1$. For further details, we refer the reader to [86].

Out-of-sample Extension. Suppose we have an unknown user $u_n \notin \mathcal{U}$ reviewing a product $\bar{p}$ that is available in our dataset. Let's denote with $\bar{r}$ the associated review. Using Bayes' rules, we can extend the model to consider users that were not included during the training phase. In this way, we can learn the latent class assignment of unseen users as follows.
Using the independence of words assumed in (4.7), we can compute $P(\bar{r}|\mathbf{y}_{u_n}^{k'}, \bar{p})$ as:

$$P(\bar{r}|\mathbf{y}_{u_n}^{k'}, \bar{p}) = \prod_{w \in \bar{r}} P(w|\mathbf{y}_{u_n}^{k'}, \bar{p}) \tag{4.20}$$

where

$$P(w|\mathbf{y}_{u_n}^{k'}, \bar{p}) = \sum_{\ell'=1}^{L} P(w|\mathbf{y}_{u_n}^{k'}, \mathbf{y}_{\bar{p}}^{\ell'}) P(\mathbf{y}_p^{\ell'}|\bar{p}). \tag{4.21}$$

Consequently, by the Bayes' rule, we have:

$$P(\mathbf{y}_{u_n}^{k'}|\bar{r}) = \frac{P(\bar{r}|\mathbf{y}_{u_n}^{k'}, \bar{p}) P(\mathbf{y}_{u_n}^{k'}|u_n)}{\sum_{k''} P(\bar{r}|\mathbf{y}_{u_n}^{k''}, \bar{p}) P(\mathbf{y}_{u_n}^{k''}|u_n)}, \tag{4.22}$$

assuming $P(\mathbf{y}_{u_n}^{k\cdot}|u_n) \sim \text{Unif}(0, 1)$ as flat prior. Given the limited information about the unknown user – we only have the words contained in their review $\bar{r}$ – we use an uninformative prior on $P(\mathbf{y}_{u_n}^{k\cdot}|u_n)$ to fully exploit the information about the available words.

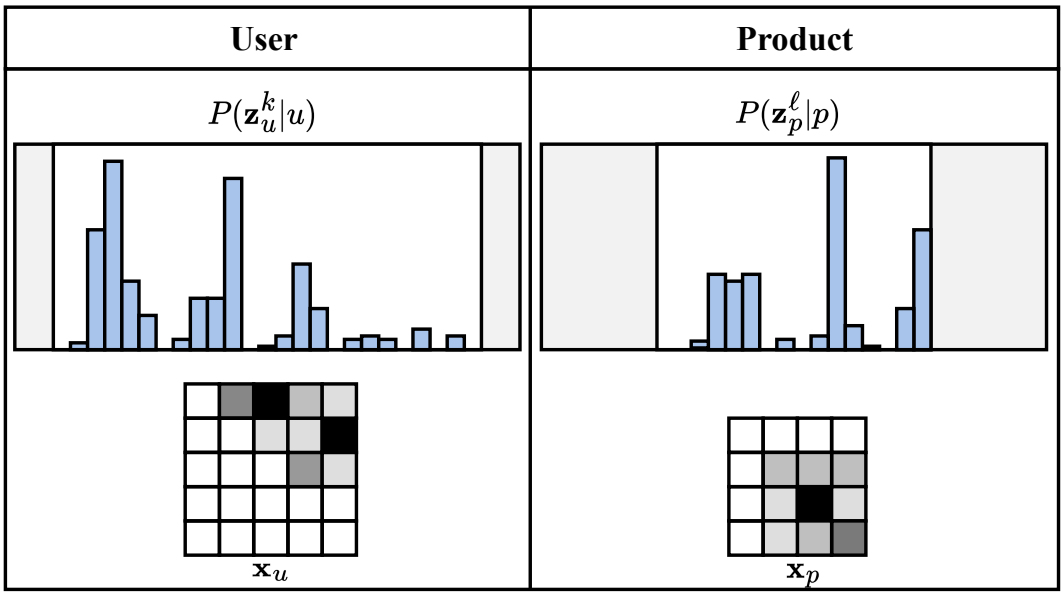


Figure 4.1: A graphical example of the conditional distributions $P(\mathbf{z}_u^k|u)$ and $P(\mathbf{z}_p^\ell|p)$ and their corresponding two-dimensional grids.

4.2.4 Rating Prediction Part

After running the EM algorithm, we have the estimates of the conditional distributions $P(\mathbf{z}_u^k|u)$ and $P(\mathbf{z}_p^\ell|p)$ for all users and products in our dataset. These quantities represent the probability assignments of the items to their respective latent classes. For the next phase of our experiment, we will use the encoded review information as input for the rating prediction task. Intuitively, given the topological organization of the latent classes on two-dimensional grids, we can think of these quantities as images where each pixel represents a latent class and the corresponding value is the latent class probability assignment. An example of the model input transformation for a sampled review is given in Figure 4.1.

Given the large employment of embedding techniques for representation learning in recommendation systems, we propose to use the learned latent class assignments as representations of users and products instead. In this way, we can generate a self-explainable latent representation of the items that, at the same time, should perform well on a rating prediction

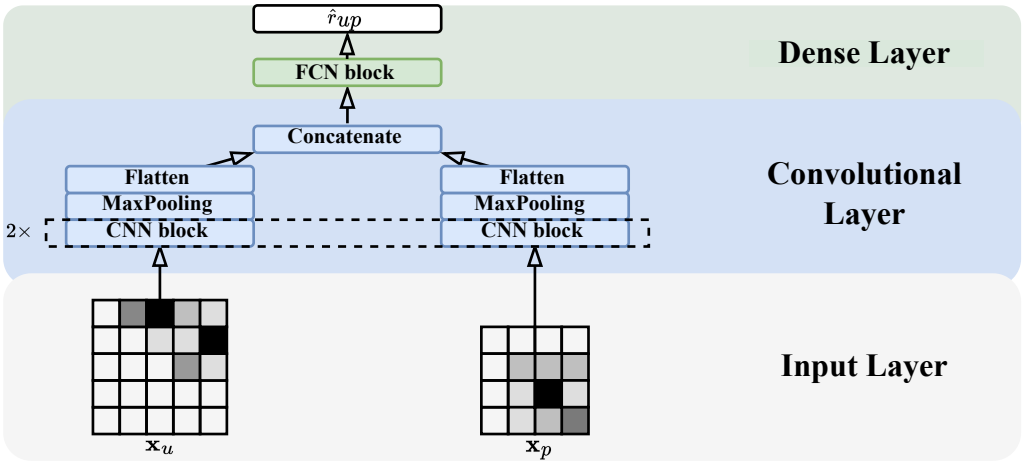


Figure 4.2: The schematic view of the proposed architecture. After the input layer, we follow the principles of related CNN-based models for rating prediction tasks. The output $\hat{r}_{up}$ represents the predicted rating for the given user-product (u,p) input.

task. The schematic view of our architecture for rating prediction is presented in Figure 4.2.

After the input layer, the CNN-block consists of the common layers used in CNN-based models. We perform two convolution operations with batch normalization and Rectified Linear Units (ReLU) [161] as activation function. As a result, the training is faster and more stable. Then, we apply a max-pooling operation to take the maximum value of each feature map obtained by the convolution operations. After the max-pooling operation, the convolutional results are reduced to a fixed size vector for both users and products. We concatenate the results given by the max pooling operation, and we use the resulting vector as input for the fully connected network (FCN) block. This block simply represents a neural network with several dense layers. More details on the structure of this block are given in Section 4.3.1. The proposed architecture follows some principles of CNN-based models for rating prediction, e.g. [87], [147]. We are not interested in getting better results than state-of-the-art approaches, instead, our ultimate goal in this part is to investigate whether our proposed model input is informative enough to get competitive results compared with more specialized techniques and architectures. In theory, one might potentially change the architecture as desired. The objective is the error of the predictions. Following previous

works, this will be validated according to the Mean Squared Error (MSE):

$$\text{MSE} = \frac{1}{|\mathcal{T}|} \sum_{(u,p)\in\mathcal{T}} (\widehat{\texttt{score}}_{u,p} - \texttt{score}_{u,p})^2 \tag{4.23}$$

where $\mathcal{T}$ represents either the test or the validation set.

4.3 Experiments

In this section, we evaluate the performance of the proposed methodology. First, we analyze the probabilistic latent model and the generated organization of user and product latent classes. Second, we investigate the generative extension of our approach. Last, we compare the performance of our model with state-of-the-art approaches for the rating prediction task.

4.3.1 Datasets and Settings

Datasets. The evaluation of the results, as in Chapter 3, is done using the *Amazon Product Data* set. We followed the same preprocessing procedure, therefore, we refer the reader to Section 3.3.1 for further details about the preprocessing steps and the statistics of the data used for training.

Experimental Settings. We set the number of user latent classes $K = 25$ and product classes $L = 16$; we evaluated the robustness of our method to changes in the hyperparameters K and L but did not observe any significant difference in both qualitative and quantitative analysis. This is in line with the constraint imposed by the topological organization; using a larger number of latent classes did not lead to overfitting. The *specificity* parameters for users and products, defined in (4.2), are set respectively to $\sigma_u = 3$ and $\sigma_p = 2$. For the SOM

initialization (Section 4.2.3), we used the Python package MINISOM.[3]

For the probabilistic model part, following previous works (e.g., [85], [86]), we apply the so-called *all but one protocol*. From each user having at least 10 reviews, we randomly select one review to be assigned to the test set. The main focus of this part of the experiment is on discovering word patterns of users and products within large collections of review data, hence we do not need a generative formulation of the model. We evaluate the training procedure according to the normalized negative log-likelihood (NLL). Based on (3.8), we have:

$$NLL_{\mathcal{T}} = -\frac{1}{|\mathcal{T}|} \sum_{(u,p,r)\in\mathcal{T}} \log \hat{P}(r|u,p) \tag{4.24}$$

where $\mathcal{T}$ represents either the training or test set.

For the rating prediction task, as in previous works, the data are randomly split by reviews into training (80%), validation (10%), and test (10%) sets. Additionally, we remove reviews from the validation and test sets if either the associated user or product does not belong to the training data set. The FCN block depicted in Figure 4.2 is a neural network with four hidden fully-connected layers $[128, 64, 32, 16]$. We evaluate different configurations of this block by changing the number of layers and the number of units for each layer. We select the best configuration based on the validation scores. The final experiment was run for 200 learning iterations and validated every iteration. A single epoch performs *Adam* optimizer [162] with a learning rate set to 0.02 and batch size of 256. The training metric is the MSE, as defined in (4.23). To prevent overfitting, we monitor the validation set score using early stopping with patience set to 10 epochs.

[3] https://github.com/JustGlowing/minisom

4.4 Results

In this section, we first visually evaluate the quality of the topographic organization of the latent class into two-dimensional grids. Then, we investigate the out-of-sample user assignments. Finally, we compare the results of the predictive task, by both considering the quantitative aspects and the interpretability capacity of the analyzed methods.

Topographic Organization of Latent Classes. Given the model parameters estimated in Section 4.2.3, we can analyze the organization of user and product latent classes by inspecting the associated word distributions. The word distribution for each user latent class, under the assumption of uninformative flat prior over latent classes, is computed as:

$$P(w|\mathbf{y}_u^{k'}) = \sum_{\ell'=1}^{L} P(w|\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}) P(\mathbf{y}_p^{\ell'}) = \frac{1}{L} \sum_{\ell'=1}^{L} P(w|\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}). \tag{4.25}$$

Analogically, the word-distribution for each product latent class can be derived as follows:

$$P(w|\mathbf{y}_p^{\ell'}) = \sum_{k'=1}^{K} P(w|\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}) P(\mathbf{y}_u^{k'}) = \frac{1}{K} \sum_{k'=1}^{K} P(w|\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}). \tag{4.26}$$

The visualization of the most probable words associated with each latent class helps us to evaluate the results. The list of the 10 most probable words for each product latent class of the *Automotive* category is provided in Table 4.1. From the results, we can notice clear patterns of the most probable words associated with each latent class. Additionally, we can observe the topological organization of the latent classes: words patterns are similar for adjacent classes in the grid. For example, adjacent classes at the top left of the grid refer to *lighting accessories*, while going down to the bottom left we can deduce that the latent classes refer to *electrical* and *oil system tools*. On the right part of the grid, instead, we have latent classes referring to *cleaning accessories*. Knowing the probabilistic assignments for a sampled item, we can easily identify the most probable latent classes and evaluate the corresponding word distributions. This visualization tool lends itself well to other types of

Table 4.1: Product-class word organization for the *Automotive* category.

light	light	blade	wax
bulb	fit	wiper	kit
bright	lead	great	product
stock	great	fit	great
white	good	good	wiper
brighter	bright	product	water
fit	price	well	snow
price	well	rain	blade
lead	order	brand	wipe
replacement	horn	jeep	rain
filter	great	product	product
oil	good	hose	paint
change	fit	great	great
fit	tool	good	pad
price	price	tank	clay
fuel	product	fit	good
socket	code	well	wax
mile	well	try	polish
wrench	purchase	say	try
good	say	quality	water
oil	good	product	product
change	great	great	jack
fluid	product	good	great
pump	mount	cover	wash
drain	item	strap	shine
price	quality	lock	spray
leak	fit	trailer	trailer
transmission	price	hitch	good
good	door	door	lift
great	hitch	step	wax
battery	plug	wheel	towel
charge	tender	leather	wash
power	battery	brush	water
charger	great	cleaner	gauge
plug	product	seat	pressure
device	cable	great	cloth
motorcycle	motorcycle	level	cleaning
phone	good	product	accurate
compressor	spark	trailer	valve
cord	winter	good	great

result investigation. For example, if we are interested in exploring a specific user latent class, one may analyze how the product latent classes change when conditioned on the user class of interest. Theoretically, the product latent classes should change according to the main *topic* of the user latent class, i.e. most of the product latent classes should now be associated with the user class of interest. The same reasoning is valid starting from a specific product latent class. Given these considerations, we provide a flexible and straightforward tool to investigate the latent characteristics of the items, making the results clear and understandable.

After evaluating the quality of the latent class organizations, we investigate whether our model is able to correctly assign unseen users to the corresponding latent classes. Given a product category, we randomly select a review $\bar{r}$ written by an unknown user u_n from the out-of-sample set. Then, given the review $\bar{r}$, we compute $P(\mathbf{y}_{u_n}^{k'}|\bar{r})$ as described in (4.22).

Out-of-sample Extension Results. Figure 4.3 shows the results taken from the *Pet Supplies* category. To make the visualization easier and intuitive, we visualize the learned user latent probability assignments over the user-class word distributions. First, by examining the most probable words associated with each user latent class, we can observe how the latent classes are topologically well organized. Then, by analyzing the words used by the unknown user listed in the bottom box, we can assume that the reviewed product is related to the *aquariums* subcategory. The results show the ability to assign the unknown user to the correct user latent classes. In fact, the most probable latent classes associated with the new user are the ones related to *aquariums*, as easily perceivable by inspecting the corresponding word lists in the grid. It is important to mention that we are able to get meaningful latent class assignments when the words used by the unknown users are, to some extent, informative. For example, if the words were only adjectives (e.g., good, nice) or verbs (e.g., look, work) it would be difficult to classify unknown users correctly. This highlights the importance of selecting a good vocabulary set as a starting point for the evaluation.

Rating Prediction and Interpretability Evaluation. We compare our method with several baselines considering both factorization-based approaches and methods that take advantage of the textual information for improving the rating prediction performance.

- **Non-negative Matrix Factorization (NMF)** is a standard baseline for CF. The rating prediction is set as the dot product between item and user factors, i.e., $q_i^\top p_u$. The latent factors are kept positive.

- **Singular Value Decomposition (SVD)** is very similar to NMF. The rating is computed as a dot product between the user and the product latent factors. In both cases, the results are poorly explainable. Indeed, it is not possible to understand user and product characteristics from the evaluation of the latent factors.

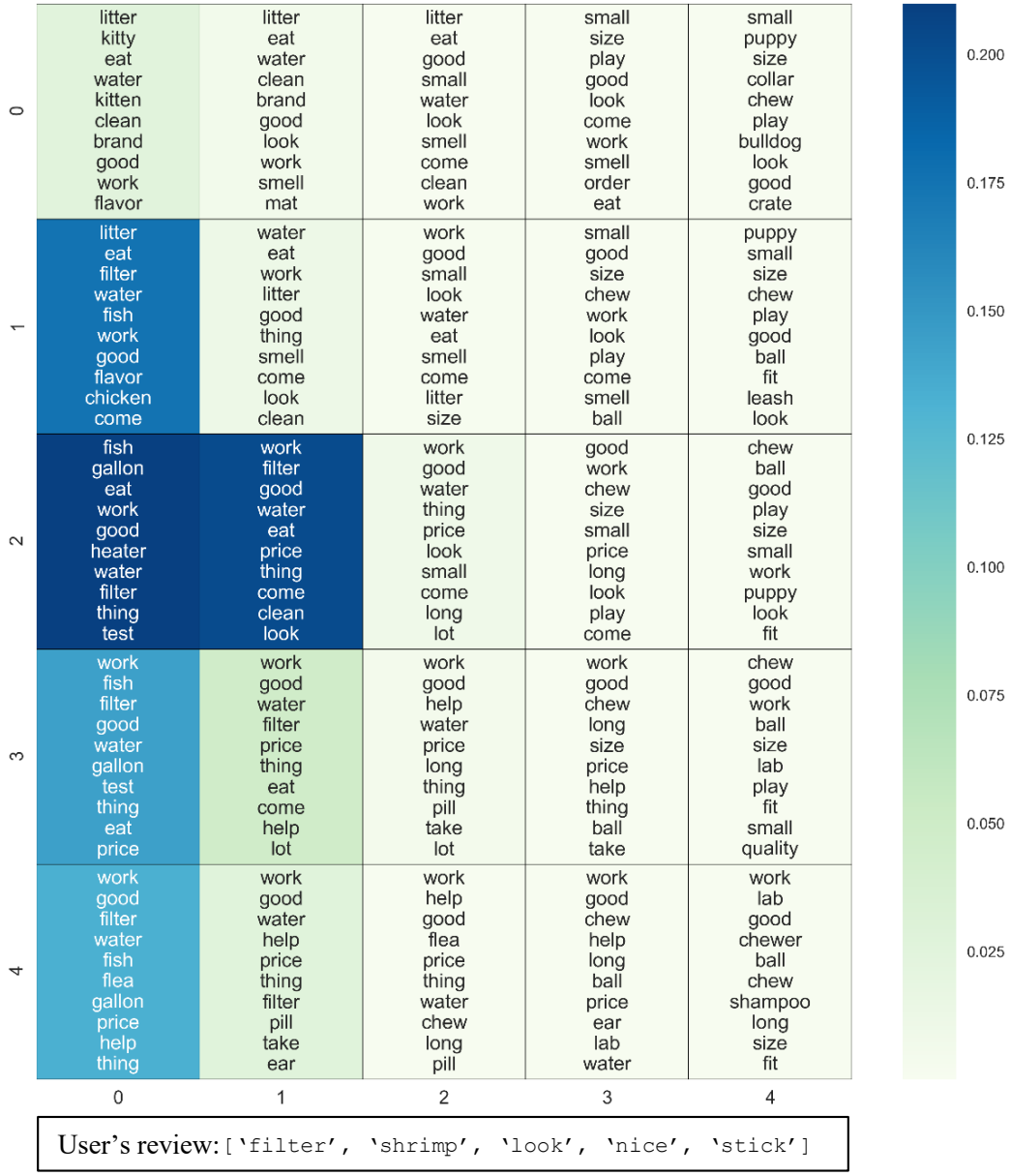


Figure 4.3: Results of the out-of-sample extension from the category *Pet Supplies*. The words in the bottom box are the ones used by the unknown user u_n to review a product p in our dataset. The considered words are included in the corresponding vocabulary $\mathcal{V}_{\text{pet}}$. Note that the darker the color, the higher is the probability assignment to the corresponding latent class.

- **Multi-Point Co-Attention Networks (MPCN)** [160] proposes a pointer-based model to extract important reviews from user and item reviews, based on the intuition that not all the reviews are equally informative. Once the important reviews are selected, a co-attentive layer learns the most important words associated. In this way, the model can learn the most informative user and product reviews for each user-item pair.

- **Deep Co-Operative Neural Networks (DCoNN)** [147] learns convolutional user and item representations from the textual information. Even though is a text-based model, the analysis is focused only on the rating prediction performances. Additionally, given the *deep* nature of the representations, the results are poorly interpretable.

- **TransNets (TNET)** [78] is an extension of the DeepCoNN model. It introduces an additional *transform* layer that learns an approximation of the review corresponding to the target user-item pair and, during the training phase, enforces it to be similar to the embedding of the actual target review. The model can be used to suggest reviews that are more similar to the one potentially written by the target user. The qualitative evaluation just rely on the visualization of some sampled reviews by highlighting the most similar sentences between the original review and the predicted most helpful one. However, if singularly evaluated, the learned embeddings do not provide any interpretable information.

- **TransRev (TR)** [93] is similar to TransNets in that it learns review embeddings as a translation of the reviewer representation to the reviewed product embedding. Ratings are predicted based on an approximation of the review embedding at test time based on the difference between the embedding of the user and the item. As in TransNets, the approximation is also used to suggest a tentative review to users for further elaborations. The evaluation of the learned word embedding is performed using t-SNE. Items and users are related to reviews by means of non-interpretable

latent representations.

- **TLCM** stands for *Transparent Latent Class Model* and represents the rating prediction model described in Section 4.2.4. This comparison helps us to investigate if the latent representations of our probabilistic model, even though not learned *ad-hoc*, represent an informative input for the rating prediction task. For the reasons explained before, our input representations, if able to get competitive results, would be better in terms of interpretability capacity compared to the embedding techniques presented in the previous works.

- **TLCM-LR** performs a simple linear regression (LR) that, for each review, takes as input the concatenation of the estimates of the corresponding user and product learned by our probabilistic model. As stated in Section 4.2.4, the architecture for rating prediction can be changed as desired. In this case, we decided to use a linear regression model since, among other methods, it is known to be a *transparent* model. Following the direction of previous investigations on the *neural hype*, this experiment allows us to understand whether the use of complicated architectures is helping to make significant progress for recommendation tasks.

As anticipated, the ultimate goal of this work is to build a review-based model suitable for both visualizing and learning user and product characteristics. Then, driven by the limitations of existing approaches to represent users and products through explainable latent vectors, we propose to exploit our estimated quantities as input for rating prediction. This idea is fully motivated by the strong correlation between ratings and reviews. In addition, we also propose to use the same information as input for an interpretable model, i.e. a linear regression, to understand its predictive performances compared to neural approaches. This experimental part completes our analysis about the *"phantom progress"* of neural-based approaches for representation learning and recommendation tasks, considering also

the interpretability capacity of the models taken into consideration.

In terms of interpretability, as already mentioned, all the baselines can provide meaningful representations in the latent space, but neither the single numbers contained in the vectors nor their dimensions have an interpretable meaning [163]. The main limitation of these techniques laid in the fact that they can capture relations among items by using vectors that are only meaningful to each other. For instance, if we try to evaluate user vectors without employing the review vectors, we would not be able to comprehend the latent textual information associated with them. Differently, our model can directly encode the textual information within the estimated quantities, providing vectors that are *self-explainable*. Additionally, our latent representations are more compact and sparse, making them easier to evaluate. Finally, the intuitive visualization properties provided by our square grids create a simple tool to investigate the results, further enhancing their interpretability.

Table 4.2: MSE for TLCM and state-of-the-art approaches.

Category	NMF	SVD	TNET	MPCN	TR	DCoNN	TLCM	TLCM-LR
Instant Videos	1.628	1.206	1.007	0.997	**0.884**	0.957	0.961	1.004
Automotive	1.171	0.818	0.946	0.861	0.855	**0.792**	0.879	0.883
Baby	2.066	1.445	1.338	1.304	1.100	1.440	**1.094**	1.309
Beauty	2.163	1.521	1.404	1.387	**1.158**	1.214	1.194	1.352
Cell Phones	2.438	1.664	1.431	1.413	**1.279**	1.378	1.294	1.447
Digital Music	1.787	1.381	1.004	0.970	0.782	0.808	0.836	1.070
Gourmet Food	1.879	1.370	1.129	1.125	**0.957**	1.542	1.064	1.117
Health	2.045	1.452	1.249	1.238	**1.011**	1.093	1.091	1.232
Musical Instr.	1.287	0.703	1.100	0.923	0.690	0.896	**0.670**	0.697
Office Products	0.988	0.719	0.840	0.779	0.724	**0.723**	0.735	0.834
Patio	1.429	0.967	1.123	1.011	**0.941**	1.020	1.045	1.100
Pet Supplies	2.071	1.438	1.346	1.328	**1.191**	1.447	1.244	1.381
Sports	1.490	1.011	0.994	0.980	**0.823**	0.898	0.898	0.958
Tools and Home	1.793	1.222	1.122	1.096	**0.879**	1.208	0.954	1.091
Toys and Games	1.542	1.111	0.974	0.973	**0.784**	0.806	0.859	0.928
Videogames	2.188	1.629	1.276	1.257	**1.082**	1.135	1.111	1.370
	1.748*	1.229*	1.143*	1.103*	0.946*	1.085*	0.996*	1.111*

In terms of rating prediction performance, we independently repeat each experiment on five different random splits and report the averaged Mean Squared Error (MSE). For a fair comparison, we conduct and compare the baselines on the same data splits, when possible, using the default configurations provided by the authors. If not, whether for difficulty in reusing the source code or its unavailability, we directly copy the results from the original papers. In detail, for SVD and NMF we used the Python package SURPRISE[4] and we selected the best learning parameters through grid search. For DeepCoNN, since the original authors' source code has not been released, we used a third-party implementation.[5] In this case, we applied the default hyperparameter setting. For the remaining baselines, we copied the results from the corresponding original papers. Results in terms of MSE are reported in Table 4.2. The asterisk indicates the macro MSE across all the product categories. From the results, in line with the conclusions of existing works, review-based methods outperform the ones based only on the numerical information. We can observe that our model is able to outperform most of the state-of-the-art approaches in the analyzed categories and has comparable performances with the best one, i.e. TransRev. Additionally, it is worth to mention that our simplest version of the architecture, i.e. TLCM-LR, is also able to get comparable results against most of the state-of-the-art approaches.

This demonstrates that: 1) by carefully encoding the textual information into the latent representations, it is possible to get similar results in comparison with more complicated techniques that provide dense and high-dimensional vector representations of the items; b) in line with the conclusions of previous investigations on CF data, also in case the textual information is used, we do not need to employ complicated neural-based models to get competitive results. The latent representations proposed in our work, even though not learned to specifically perform well on a rating prediction task, are able to maintain competitiveness in comparison with more specialized architectures. This clearly suggests that one may po-

[4] http://surpriselib.com/

[5] https://github.com/chenchongthu/DeepCoNN

tentially use the proposed model input with nice interpretable properties, without worsening the rating prediction task considerably.

4.5 Summary

Recently, several approaches for rating predictions of textual reviews in the framework of deep neural networks have appeared in the literature [78], [89], [92], [93], [147], [160]. Given the highly stochastic nature of the data and relative data sparsity, one can legitimately ask to what extent can the full predictive power of deep networks be utilized in this context. The question is even more relevant when one realises that clear interpretability of the deep network functionality is still an open problem [8]. To answer this question, we present an approach for product rating prediction using a relatively simple and interpretable latent class probabilistic model utilizing topographic organization of user and product latent classes based on the review information. In existing works, the review information is usually exploited to enhance the rating prediction performances, but is not fully inspected to understand user and product features. In contrast, we propose a deeper understanding of the data, presenting a two-step approach that starts from the interpretable organization of textual information to arrive at the rating prediction task. The organization of the latent classes on 2-dimensional grids provides a visualization tool that can be used to statistically investigate user and product features from a review-based perspective. Through this organization, we can arrange complicated and unstructured textual data in a simple way. The thorough analysis in the experimental section demonstrates the ease of analyzing the latent review patterns using tools from probabilistic theory. The visualization of the results, presented in Section 4.4, shows that the lower-dimensional latent representations of users and products are a good approximation of the textual input space. Consequently, driven by the assumption that ratings and reviews are strongly correlated, we propose to use the resulting

latent features as input for a rating prediction task. In this part, we contribute to the debate about the *"phantom progress"* of deep learning approaches for recommendation tasks. Our investigation, differently from the previous ones, is mainly focused on methods that take advantage of the textual information for rating prediction tasks, and includes an evaluation that considers the capacity of such models to represent users, products, and reviews utilizing interpretable latent vectors. The results suggest that, also in the textual-based case, the use of dense and complicated representations is not fully motivated. Indeed, even though our representations are not learned for a rating prediction task specifically, the results are comparable to models that learn *ad-hoc* representations. Nonetheless, being highly interpretable, the proposed latent representations overcome the limitations of the common embedding techniques used in most of the considered previous works. Finally, the prediction results of our linear regression model suggest that we do not need to implement deep architectures either. This simple model is able to outperform some of the baselines and get comparable results with the remaining ones, while being fully transparent. Naturally, there is always a trade-off between model capabilities and interpretability. The nice and explainable visualization properties of our constrained model may affect the modeling capabilities for rating prediction. On the other hand, better modeling capabilities through common embedding techniques and deep architectures provide representations that are created for performing well on a specific task, but at the price of losing the human interpretation of the results.

Chapter Five

Learning to Explain Graph Black-box Models

5.1 Motivations

In the previous chapters, we have seen how to instill interpretable concepts into representation learning tasks using textual information. The proposed solutions help investigate the features the model thinks are relevant for prediction and create a 'translation' of the numerical representations to human-understandable information. However, oftentimes, we do not have a source of intelligible information to exploit for learning *self-interpretable* vectors. In these cases, to increase human trust in ML-based models, we should try to explicitly discover the inner reasoning of the black-box models by using *self-interpretable models*. Since graphs occur naturally in several domains such as chemistry, biology, recommender systems (as in the previous chapters), and medicine, Graph Neural Networks (GNNs) have experienced widespread adoption. Following a trend towards building more interpretable machine learning models, there have been numerous recent proposals to provide explanations for GNNs. Most of the existing approaches, as seen in Chapter 2, provide post-hoc explanations starting

from an already trained GNN to identify edges and node attributes that explain the model's prediction. However, as highlighted in Faber, K. Moghaddam, and Wattenhofer [46], there might be some discrepancy between the ground-truth explanations and those attributed to the trained GNNs. Indeed, post-hoc explanations are often not able to faithfully represent the mechanisms of the original model [164]. Furthermore, a recent work has also shown that post-hoc attribution methods are often not better than random baselines on the standard evaluation metrics for explanation accuracy and faithfulness [129]. Unfortunately, as discussed in Section 2.4.4, the very definition of what constitutes a faithful explanation is still open to debate and there exist several competing positions on the matter. To recall the definition of *faithfulness* we introduced in Chapter 2 that we will use in this methodological part, an explanation is considered to be faithful if (i) it is a significantly smaller subgraph of the input graph and (ii) we know that *only its structure* is used in the message-passing operations for the prediction.

To overcome the limitations of post-hoc methods, a recently proposed alternative is the learning to explain (L2X) paradigm [12]. The core difference to post-hoc methods is that the models are trained to, in the forward pass, discretely select a small subset of the input features as well as the parameters of a downstream model that uses only the selected features to make a prediction. The selected features are, therefore, faithful by design as they are the only ones used by the downstream model. Since the subset of features is sampled discretely, L2X requires a method for computing gradients of an expectation over a discrete probability distribution. [12] proposed a gradient estimator based on a relaxation of the discrete samples and tailored to the k-subset distribution. However, since the original work only considers the case of selecting *exactly-k* features, directly applying prior methods to graph learning tasks is not possible and requires significant changes. Thus, since prior work's gradient estimators do not work with arbitrary optimization problems but are restricted to the k-subset distribution, using the L2X paradigm for graphs is highly non-trivial.

With this work, we bring the L2X paradigm to graph representation learning to address the problems related to graph-based (black-box) models and GNN post-hoc explainers. In the original paper, the instance-wise feature selector in defined as a model which returns a distribution over the subset of features given the input vector. Likewise, to adapt the idea to work for graphs, we have to assume to work on a distribution of graphs where – given the matrix of edge weights (unnormalized probabilities) $\boldsymbol{\theta}$ – we sample a subset of edges (i.e., a new adjacency matrix) $\boldsymbol{Z}$ from the set of feasible solutions $\mathcal{F}$ of the chosen optimization problem. The important ingredient is a recently proposed method for computing gradients of an expectation over a complex exponential family distribution [165]. The method facilitates approximate gradient backpropagation for models combining continuously differentiable GNNs with a black-box solver of combinatorial problems defined on graphs. Crucially, this allows us to learn to sample subgraphs with beneficial properties such as being connected and sparse. Contrary to prior work, this also creates a dependency between the random variables representing the presence of edges. The proposed framework L2xGnn, therefore, learns to select explanatory subgraph motifs and uses *these and only these motifs* for its message-passing operations. To the best of our knowledge, this is the first method for learning to explain GNNs. The proposed framework is extensible as it can work with any optimization algorithm for graphs imposing properties on the sampled subgraphs.

We compare two different sampling strategies for obtaining sparse subgraph explanations resulting from two optimization problems on graphs: (1) the maximum-weight k-edge subgraph and (2) the maximum-weight k-edge connected subgraph problem. We show empirically that L2xGnn when combined with a base GNN does not lose accuracy on several benchmark datasets. Moreover, we evaluate the explanations quantitatively and qualitatively. We also analyze the ability of L2xGnn to help in detecting shortcut learning which can be used for debugging the GNN. Given the characteristics of the proposed method, our work improves model interpretability and increases the clarity of known black-box models as GNNs while

maintaining competitive predictive capabilities.

5.2 Model Definition

We propose a method that learns both (i) the parameters of a graph generative model and (ii) the parameters of a GNN operating on sparse subgraphs approximately sampled from said generative model in the forward pass. In line with prior work on learning to explain [12], the maximum probability subgraph is then used at test time to make the prediction and, therefore, serves as the faithful explanation. Since we aim to sample graphs with certain properties (e.g., connected subgraphs) we need a new approach to sampling and gradient estimation. Contrary to prior work on edge masking [110] which treats edges as independent binary random variables, we use a recently introduced method for backpropagating through optimization algorithms. This allows us to select subgraphs with specific properties and, therefore, to explicitly model dependencies between edge variables. Intuitively, our approach consists of three main components. In the first part, an upstream model $h_{\boldsymbol{v}}$ learns the edge weights $\theta_{(i,j)}$ for each edge (i, j) belonging to the given input graph. In the subsequent component, the learned edge matrix $\boldsymbol{\theta}$ is given as input to an optimization algorithm `opt`. The algorithm considers the weights $\boldsymbol{\theta}$ as unnormalized probabilities to sample discretely a new adjacency matrix $\boldsymbol{Z}$. Finally, the resulting sampled subgraph $\boldsymbol{z}$ is used in the last component, the downstream model $f_{\boldsymbol{u}}$, to make the final prediction. A graphical representation of our approach is presented in Figure 5.1. Considering the proposed workflow, we can identify two main challenges related to our method: a) how to learn $\boldsymbol{\theta}$ such that we can improve the selection of the subgraph $\boldsymbol{z}$; b) how to estimate and backpropagate the gradient through a discrete component (i.e., `opt`). In the following subsections, we will explain our framework in more detail and provide technical solutions for the introduced challenges. In Subsection 5.2.1, we formalize the problem and describe rigorously our framework. In Subsection 5.2.2,

we describe the gradient estimation method used in this work. Finally, in Subsection 5.2.3, we detail how to use and adapt the introduced concepts to work explaining GNNs.

5.2.1 Problem Statement and Framework

We aim to jointly learn the parameters of a probability distribution over subgraphs *with certain properties* and the parameters of a GNN operating on graphs sampled from said distribution in the context of the graph classification problem. Here, the training data consists of a set of triples $\{(\mathbf{A}, \mathbf{X}, \mathbf{y})_j\}, j \in \{1, ..., N\}$, where $\mathbf{A}$ is an $n \times n$ binary adjacency matrix, $\mathbf{X} \in \mathbf{R}^{n \times d}$ a node attribute matrix with d the number of node attributes, and $\mathbf{y}$ the target graph label. First, we have a learnable function $h_{\boldsymbol{v}} : \mathcal{A} \times \mathcal{X} \rightarrow \Theta$ where $\mathcal{A}$ is the set of all $n \times n$ adjacency matrices, $\mathcal{X}$ the set of all attribute matrices, $\boldsymbol{v}$ are the parameters of h, and Θ the set of possible edge parameter values. The function, which we refer to as the upstream model, maps the adjacency and attribute matrix to a matrix of edge weights $\boldsymbol{\theta} \in \mathbb{R}^{n \times n}$. Intuitively, $\boldsymbol{\theta}_{i,j}$ is the prior probability of edge (i, j).

Next, we assume an algorithm $\texttt{opt} : \Theta \rightarrow \mathcal{A}$ which returns the (approximate) solutions to an optimization problem on edge-weighted graphs. Examples of such optimization problems are the maximum-weight spanning tree or the maximum-weight k-edge connected subgraph problems. The optimization algorithm is treated as a black box. One can choose the optimization problem according to the application's requirements. We have found, for instance, that the connected subgraphs lead to better explanations in the domain of chemical compound classification. Contrary to prior work, the optimization problem creates a dependency between the binary variables modeling the edges.

For every binary adjacency matrix $\boldsymbol{Z} \in \mathcal{A}$, we write $\boldsymbol{Z} \in \mathcal{F}$ if and only if the adjacency matrix is a feasible solution (not necessarily an optimal one) of the chosen optimization

problem. We can now define a discrete exponential family distribution as

$$p(\boldsymbol{Z};\boldsymbol{\theta}) = \begin{cases} \exp\left(\langle \boldsymbol{Z}, \boldsymbol{\theta} \rangle_F - B(\boldsymbol{\theta})\right) & \text{if } \boldsymbol{Z} \in \mathcal{F}, \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product and $B(\boldsymbol{\theta})$ is the log-partition function defined as

$$B(\boldsymbol{\theta}) = \log \left(\sum_{\boldsymbol{Z} \in \mathcal{F}} \exp\left(\langle \boldsymbol{Z}, \boldsymbol{\theta} \rangle_F\right) \right).$$

Hence, p is a probability distribution over adjacency matrices that are feasible solutions to the optimization problem under consideration. Each feasible adjacency matrix's probability mass is proportional to the product of its edge weights. For example, if the optimization problem is the maximum-weight k-edge connected subgraph problem, the distribution assigns a non-zero probability mass to all adjacency matrices of graphs that have k edges and are connected.

Given an optimization problem, we would like to sample exactly from the above probability distribution $p(\boldsymbol{Z};\boldsymbol{\theta})$. Unfortunately, this is intractable since computing the log-partition function is in general NP-hard. However, as in prior work [165], we can use perturb-and-MAP [166] to *approximately* sample from the above distribution as follows. Let $\boldsymbol{\epsilon} \sim \rho(\boldsymbol{\epsilon})$ be a $n \times n$ matrix of appropriate random variables such as those following the Gumbel distribution. We can then *approximately* sample an adjacency matrix $\boldsymbol{Z}$ from $p(\boldsymbol{Z};\boldsymbol{\theta})$ by computing

$$\boldsymbol{Z} = \texttt{opt}(\boldsymbol{\theta} + \boldsymbol{\epsilon}).$$

Hence, we can approximately sample by perturbing the edge weights (unnormalized probabilities) $\boldsymbol{\theta}$ and by applying the optimization algorithm to these perturbed weights.

In the final part of the model (the downstream model) we use the sampled $\boldsymbol{Z}$ as the input adjacency matrix to a message-passing neural network $f_{\boldsymbol{u}} : \mathcal{A} \times \mathcal{X} \to \mathcal{Y}$ computing $\hat{\boldsymbol{y}} = f_{\boldsymbol{u}}(\boldsymbol{Z}, \boldsymbol{X})$.

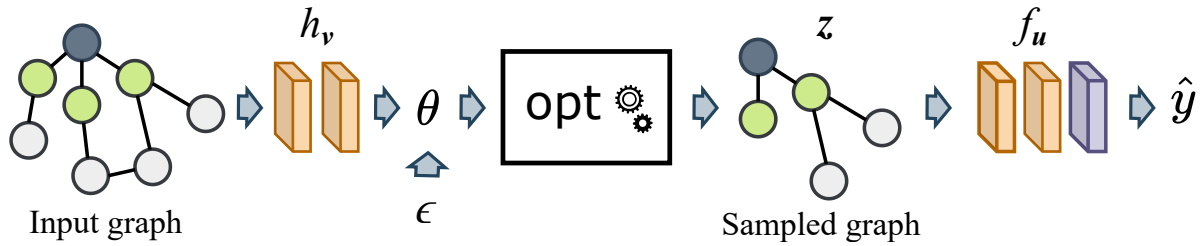


Figure 5.1: Workflow of the proposed approach. The upstream model h_v learns to assign weights $\theta_{.,.}$ for each edge in the input graph. The edge matrix $\boldsymbol{\theta}$ – perturbed with $\boldsymbol{\epsilon}$ – is then utilized as input by the optimization algorithm $\texttt{opt}$ to sample a subgraph $\boldsymbol{z}$ with specific characteristics. Finally, the downstream model f_u uses *only* the information about the sampled (sub)graph to make a prediction.

In summary, we have the following model architecture for training input data $(\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{y})$:

$$\boldsymbol{\theta} = h_v(\boldsymbol{A}, \boldsymbol{X}) \qquad \text{with} \quad \boldsymbol{A} \in \mathcal{A}, \boldsymbol{X} \in \mathcal{X}, \tag{5.2}$$

$$\boldsymbol{Z} = \texttt{opt}(\boldsymbol{\theta} + \boldsymbol{\epsilon}) \qquad \text{with} \quad \boldsymbol{\epsilon} \sim \rho(\epsilon), \boldsymbol{\epsilon} \in \mathbb{R}^{n \times n}, \tag{5.3}$$

$$\hat{\boldsymbol{y}} = f_u(\boldsymbol{Z}, \boldsymbol{X}) \qquad \text{with} \quad \hat{\boldsymbol{y}} \in \mathcal{Y}, f_u : \mathcal{A} \times \mathcal{X} \to \mathcal{Y}. \tag{5.4}$$

Figure 5.1 illustrates the architecture. With $\boldsymbol{\omega} = (\boldsymbol{u}, \boldsymbol{v})$ the learnable parameters of the model and the target variable $\boldsymbol{y}$ the loss is now defined as:

$$L(\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\omega}) = \mathbb{E}_{\boldsymbol{\epsilon} \sim \rho(\epsilon)}[\ell(f_u(\boldsymbol{Z}, \boldsymbol{X}), \boldsymbol{y})], \tag{5.5}$$

with $\boldsymbol{Z} = \texttt{opt}(\boldsymbol{\theta} + \boldsymbol{\epsilon})$, $\boldsymbol{\theta} = h_v(\boldsymbol{A}, \boldsymbol{X})$, and $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ a point-wise loss function. The gradient of L with respect to $\boldsymbol{u}$ is

$$\nabla_u L(\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\omega}) = \mathbb{E}[\partial_u f_u(\boldsymbol{Z}, \boldsymbol{X})^\intercal \nabla_y \ell(\hat{\boldsymbol{y}}, \boldsymbol{y})]$$

which can be estimated by Monte-Carlo sampling. In contrast, the gradient of L with respect to $\boldsymbol{v}$ is:

$$\nabla_v L(\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\omega}) = \partial_v h_v(\boldsymbol{A}, \boldsymbol{X})^\intercal \nabla_{\boldsymbol{\theta}} L(\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\omega}),$$

where the challenge is to estimate $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\omega}) = \nabla_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{\epsilon} \sim \rho(\epsilon)}[\ell(f_u(\boldsymbol{Z}, \boldsymbol{X}), \boldsymbol{y})]$ because $\boldsymbol{Z} = \texttt{opt}(\boldsymbol{\theta} + \boldsymbol{\epsilon})$ is not continuously differentiable woth respect to $\boldsymbol{\theta}$. While it would be possible to use the score function estimator, its high variance typically makes it less competitive in practice [165].

5.2.2 Implicit Maximum-Likelihood Learning

The variant of I-MLE we use in this work estimates $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\omega})$ by implicitly creating a target distribution $q(\boldsymbol{Z}; \boldsymbol{\theta}')$ using perturbation-based implicit differentiation [167]. Here, the parameters $\boldsymbol{\theta}$ are moved in the direction of $-\nabla_{\boldsymbol{Z}} \ell(f_{\boldsymbol{u}}(\boldsymbol{A}, \boldsymbol{X}), \boldsymbol{y}))$, the negative gradient of the downstream loss with respect to the sampled adjacency matrix $\boldsymbol{Z}$, to construct $\boldsymbol{\theta}'$

$$q(\boldsymbol{Z}; \boldsymbol{\theta}') := p(\boldsymbol{Z}; \boldsymbol{\theta} - \lambda \nabla_{\boldsymbol{Z}} \ell(f_{\boldsymbol{u}}(\boldsymbol{Z}, \boldsymbol{X}), \boldsymbol{y})) \tag{5.6}$$

with $\boldsymbol{Z} = \texttt{opt}(\boldsymbol{\theta} + \boldsymbol{\epsilon})$ and $\lambda > 0$ the strength of the perturbation. Intuitively, by moving the weights $\boldsymbol{\theta}$ into the direction of the negative gradients of $\boldsymbol{Z}$, the resulting distribution q is more likely to generate samples with a lower downstream loss. We approximate $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\omega})$ with a single-sample Monte Carlo estimate of the gradients of the KL divergence between p and q:

$$\nabla_{\boldsymbol{\theta}} L(\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\omega}) \approx \frac{1}{\lambda} \left(\texttt{opt}(\boldsymbol{\theta} + \boldsymbol{\epsilon}) - \texttt{opt}(\boldsymbol{\theta}' + \boldsymbol{\epsilon}) \right). \tag{5.7}$$

In other words, $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{y}; \boldsymbol{\omega})$ is approximated by the difference between an approximate sample from $p(\boldsymbol{Z}; \boldsymbol{\theta})$ and an approximate sample from $q(\boldsymbol{Z}; \boldsymbol{\theta}')$. In this way we move the distribution $p(\boldsymbol{Z}; \boldsymbol{\theta})$ closer to $q(\boldsymbol{Z}; \boldsymbol{\theta}')$. For further details on I-MLE we refer the reader to the original paper [165].

5.2.3 L2xGnn: Learning to Explain GNNs with I-MLE

We now describe the class of L2xGnn models we use in the experiments. First, we need to define the function $h_{\boldsymbol{v}}(\boldsymbol{A}, \boldsymbol{X})$. Here we use a standard GNN (see equation 2.1) to compute for every node i and every layer ℓ the vector representation $\mathbf{h}_i^{\ell} = h_{\boldsymbol{v}}(\boldsymbol{A}, \boldsymbol{X})_{i,1:d}$. We then compute the matrix of edge weights by taking the inner product between each pair of node embeddings. More formally, we compute $\boldsymbol{\theta}_{i,j} = \langle \mathbf{h}_i^{\ell}, \mathbf{h}_j^{\ell} \rangle$ for some fixed ℓ. Typically, we choose $\ell = 1$.

Algorithm 1 Greedy algorithm $\texttt{opt}_{\texttt{con}}$ for the maximum-weight k-edge connected subgraph problem.

Input:

Input graph $\mathcal{G} = (V, E)$

Number of edges k

Edge weights $\boldsymbol{\theta}$

Initialize:

$e = \arg\max_{e_{i,j}} \boldsymbol{\theta}_{i,j}$

Set of selected edges $S = \{e\}$

Set of edges adjacent to selected edge $N = \mathcal{N}(e)$

while $|S| < k$ and $|N| > 0$ **do**

$e = \arg\max_{e_{i,j} \in N} \boldsymbol{\theta}_{i,j}$

$S = S \cup \{e\}$

$N = N \cup \mathcal{N}(e)$

$N = N - S$

end while

Return: Adjacency matrix $\boldsymbol{Z}$ of the subgraph induced by the set of selected edges S.

In this work, we sample the noise perturbations $\boldsymbol{\epsilon}$ from the sum of Gamma distribution [165]. Other noise distributions such as the Gumbel distribution are possible.

Sampling Constrained Subgraphs. An advantage of the proposed method is its ability to integrate any graph optimization problem as long as there exists an algorithm `opt` for computing (approximate) solutions. In this work, we focus on two optimization problems: (1) The maximum-weight k-edge subgraph and (2) the maximum-weight k-edge connected subgraph problems. The former aims to find a maximum-weight subgraph with k edges. The latter aims to find a *connected* maximum-weight subgraph with k edges. Other optimization problems are possible but we found that sparse and connected subgraphs provide a good

efficiency-effectiveness trade-off.

Computing maximum weight k-edge subgraphs is highly efficient as we only need to select the k edges with the maximum weights. In order to compute *connected* k-edge subgraphs we use a greedy approach. First, given a number k of edges, we select a single edge $e_{i,j}$ with the highest weight $\boldsymbol{\theta}_{i,j}$ from the input graph. At every iteration of the algorithm, we select the next edge such that it (a) is connected to a previously selected edge and (b) has the maximum weight among all those connected edges. A more detailed description of the greedy algorithm is given in Algorithm 1.

Finally, we need to define the function $f_{\boldsymbol{u}}$ (the downstream function) of the proposed framework. Here, we again use a message-passing GNN that follows the update rule

$$\mathbf{h}_i^{\ell} = \gamma\left(\mathbf{h}_i^{\ell-1}, \square_{j \in \mathcal{N}(v_i)} \phi\left(\mathbf{h}_i^{\ell-1}, \mathbf{h}_j^{\ell-1}, r_{ij}\right)\right). \tag{5.8}$$

The neighborhood structure $\mathcal{N}(\cdot)$, however, is defined through the output adjacency matrix $\boldsymbol{Z}$ of the optimization algorithm `opt`

$$j \in \mathcal{N}(v_i) \iff \boldsymbol{Z}_{i,j} = \boldsymbol{Z}_{j,i} = 1. \tag{5.9}$$

Hence, if after the subgraph sampling, there exists a node v_i which is an isolated node in the adjacency matrix $\boldsymbol{Z}$, that is, $\boldsymbol{Z}_{i,j} = \boldsymbol{Z}_{j,i} = 0\ \forall j \in \{1, ..., n\}$, the embedding of the node will not be updated based on message passing steps with neighboring nodes. This means that, for *isolated* nodes, the only information used in the downstream model is the one from the nodes themselves. Conceptually, $\boldsymbol{Z}$ works as a mask over the messages m_{ij}^{ℓ} computed at each layer ℓ.

The adjacency matrix $\boldsymbol{Z}$ is then used in all subsequent layers of the GNN. In particular, for one layer ℓ we have

$$\mathbf{H}_{\ell} = \textsc{Gnn}_{\ell}(\boldsymbol{A} \odot \boldsymbol{Z}, \mathbf{H}_{\ell-1}), \tag{5.10}$$

where $\odot$ is the Hadamard product. Finally, the remaining part of the L2xGnn network for the graph classification is

$$\mathbf{h}_G = \text{Pool}(\mathbf{H}_\ell) \qquad \hat{\boldsymbol{y}} = \eta(\mathbf{h}_G), \tag{5.11}$$

where we use a global pooling operator to generate the (sub)graph representation $\mathbf{h}_G$ that will then be used by the MLP network $\eta(\cdot)$ to output a probability distribution $\hat{\boldsymbol{y}}$ over the class labels. Finally, a loss function is applied whose gradients are used to perform back-propagation. At test time, we use the maximum-probability subgraph for the explanation and prediction, that is, we do not perturb at test time.

5.3 Experiments

In this section, we analyze the performance of the proposed approach. First, we evaluate the predictive performance of the model compared to baselines. Second, we analyze the explanatory subgraphs for datasets for which we know the ground-truth motifs. Finally, we analyze whether the generated output can be helpful for model debugging purposes. We also report several ablation studies to investigate the effects of different model choices on the results. For the remainder of the manuscript, we use L2xGnn$_{\texttt{dsc}}$ and L2xGnn for referring to the maximum-weight k-edge subgraph and to the maximum-weight k-edge *connected* subgraph problem respectively.

5.3.1 Datasets and Settings

Datasets. To understand the change in the predictive capabilities of the base models when integrating L2xGnn, we use six real-world datasets for graph classification tasks: MUTAG [168], PROTEINS [169], YEAST [170], IMDB-BINARY, IMDB-MULTI [171], and DD [172]. In Table 5.1, we report the statistics of the datasets used for graph classification

tasks. We use datasets from different domains including biological data, social networks, and bioinformatics data. For a comprehensive evaluation, we include datasets with different characteristics, such as a larger number of graphs or a larger number of nodes and edges.

To quantitatively evaluate the quality of the explanations, we use datasets that include

Table 5.1: Statistics of the datasets.

Number of	Nodes (avg)	Edges (avg)	Graphs	Classes
DD	284.32	715.66	1178	2
MUTAG	17.93	19.79	188	2
IMDB-B	19.77	96.53	1000	2
IMDB-M	13.00	65.94	1500	3
PROTEINS	39.06	72.82	1113	2
YEAST	21.54	22.84	79601	2

ground-truth edge masks. In particular, we use MUTAG_0 and BA2Motifs. MUTAG_0 is a dataset introduced in Tan, Geng, Fu, *et al.* [115] which contains the benzene-NO_2 (i.e., a carbon ring with a nitro group (NO_2) attached) as the only discriminative motif between positive and negative labels. BA2Motifs is a synthetic dataset that was first introduced in Luo, Cheng, Xu, *et al.* [107]. The base graphs are Barabasi-Albert (BA) graphs. 50% of the graphs are augmented with a *house-motif* graphs, the rest with a *5-node cycle motif*. The discriminative subgraph leading to different predictions is the motif attached to the BA graph.

Experimental Settings. To evaluate the quality of our approach, we use L2xGnn with several GNN base models including GCN [30], GIN [22] and GraphSAGE [20]. We compare the results when using the original model and when the same model is combined with our XAI method. For model selection and evaluation, to fairly compare the methods, we follow a previously proposed protocol[1]. We perform a 10-fold cross validation where the hyperparameter selection is done according to the validation accuracy. The selection is performed

[1] https://github.com/pyg-team/pytorch_geometric/tree/master/benchmark/kernel

for the number of layers (L) $[1, 2, 3, 4]$ and the number of hidden units (H) $[16, 32, 64, 128]$. For both parameters, the selected numbers represent a standard range of values to decide the characteristics of the backbone architecture. For a fair comparison with the backbone architectures, we select the best configuration for each dataset, and we integrate our approach into the best model. Instead of fixing a value k for each input graph, we compute k based on a ratio R of edges to be kept. Once the hyperparameters of the default model are found, we select the best ratio R (in terms of percentage of edges to keep) from the set of values $[0.4, 0.5, 0.6, 0.7]$ based again on the validation accuracy. We do not include extreme values for two reasons: (1) smaller values for R lead to reduced predictive capabilities and not meaningful explanatory subgraphs; and (2) higher values would not remove enough edges compared to the original input. Finally, we choose the perturbation intensity λ from the values $[10, 100, 1000]$ taken from the original paper [165].

Hyperparameter Selection. Experiments were run on a single Linux machine with Intel Core i7-11370H @ 3.30GHz, 1 GeForce RTX 3060, and 16 GB RAM. The best hyperparameter configuration for each model and dataset used for graph classification tasks is reported in Table 5.2.

Table 5.2: Hyperparameter settings for graph classification tasks. H and L represent the number of hidden units and the number of layers respectively.

Dataset	GCN				GIN				GraphSAGE			
	H	L	R	λ	H	L	R	λ	H	L	R	λ
DD	128	2	0.6	10	64	1	0.5	100	128	1	0.6	100
MUTAG	128	3	0.6	1000	128	4	0.5	10	128	3	0.4	10
IMDB-B	128	3	0.4	100	64	3	0.4	1000	64	1	0.4	10
IMDB-M	64	3	0.6	10	128	4	0.6	10	64	1	0.4	100
PROTEINS	128	3	0.7	100	128	4	0.5	10	64	3	0.4	10
YEAST	128	3	0.6	10	128	3	0.6	10	32	4	0.5	100

Table 5.3: Prediction test accuracy (%) for graph classification tasks over ten runs.

Method	Dataset					
	DD	MUTAG	IMDB-B	IMDB-M	PROTEINS	YEAST
GCN	**72.0** ± 2.4	73.4 ± 8.3	73.1 ± 3.2	50.0 ± 2.8	71.8 ± 4.4	88.1 ± 0.1
L2xGcn$_{dsc}$	71.9 ± 3.1	73.9 ± 11.1	66.0 ± 5.4	**50.3** ± 3.2	71.1 ± 3.4	**88.2** ± 0.2
L2xGcn	71.9 ± 3.6	**74.5** ± 8.2	**73.4** ± 4.7	49.0 ± 2.2	**72.0** ± 5.3	88.1 ± 0.1
GIN	72.2 ± 2.7	**82.7** ± 5.1	72.1 ± 5.0	**49.0** ± 4.7	70.8 ± 4.5	**88.3** ± 0.1
L2xGin$_{dsc}$	**73.9** ± 5.1	81.4 ± 9.2	65.0 ± 5.0	48.8 ± 3.2	68.5 ± 2.9	88.2 ± 0.1
L2xGin	72.0 ± 3.0	82.5 ± 7.8	**72.4** ± 4.5	47.9 ± 3.5	**70.9** ± 3.4	88.0 ± 0.2
GraphSage	72.1 ± 3.9	73.4 ± 7.5	72.2 ± 4.8	50.7 ± 3.7	71.3 ± 5.1	**88.2** ± 0.1
L2xGsg$_{dsc}$	**72.7** ± 3.8	75.1 ± 7.7	**73.8** ± 2.8	50.6 ± 3.2	**71.3** ± 4.1	88.0 ± 0.1
L2xGsg	72.5 ± 3.9	**79.8** ± 8.1	73.0 ± 4.1	**50.8** ± 2.7	70.7 ± 4.6	88.1 ± 0.2

5.4 Results

Graph Classification Comparison with Base GNNs. Following the experimental procedure proposed in Zhang, Liu, Wang, *et al.* [125], Table 5.3 lists the results of using L2xGnn with base GNN architectures for graph classification tasks. The primary goal of this work is not to provide a better predictive model, but to provide faithful explanation masks while maintaining similar predictive performance. This analysis is important since inherent interpretable networks are known for creating a trade-off with the predictive capabilities of the model, and practitioners may not be willing to sacrifice the prediction accuracy for increased transparency [173]. Nevertheless, we observe that L2xGnn is competitive and often even outperforms the base GNN models on the benchmark datasets.

Explanation Accuracy. We compare the proposed method with popular post-hoc explanation techniques including the GNN-Explainer [106], PGE-Explainer [107], GradCAM [103], GNN-LRP [122], and SubgraphX [111][2]. We train a 3-layer GIN for 200 epochs with hidden dimensions equal to 64 and a learning rate equal to 0.001. We save the best model

[2]Implementations taken from the Dig library [174].

Table 5.4: Evaluation of explanation accuracy (%) on synthetic graph classification datasets using a 3-layer GIN architecture. The lowest standard deviation for each metric is underlined. With the exception of L2xGnn, none of the approaches can guarantee *faithful* explanations where the explanation is exclusively used during message passing operations

Dataset	BA-2MOTIFS				$MUTAG_0$			
	Acc.	Pr.	Rec.	F_1	Acc.	Pr.	Rec.	F_1
GNN-Exp.	44.6 ± 2.4	22.7 ± 0.9	62.9 ± 3.3	32.9 ± 1.0	47.4 ± 2.3	42.2 ± 2.4	69.2 ± 2.4	50.2 ± 2.1
GradCAM	77.1 ± 11.5	50.1 ± 16.1	72.4 ± 23.2	59.0 ± 19.0	**78.0** ± 1.3	**85.6** ± 2.8	60.8 ± 3.9	68.8 ± 2.2
PGE-Exp.	36.7 ± 18.9	17.5 ± 5.9	66.6 ± 22.5	27.7 ± 9.4	65.0 ± 9.6	57.3 ± 12.3	54.7 ± 12.2	54.9 ± 12.0
GNN-LRP	77.3 ± 2.5	34.3 ± 16.8	36.4 ± 19.7	33.0 ± 15.7	71.7 ± 7.3	78.6 ± 9.2	43.5 ± 16.7	53.4 ± 17.1
SubgraphX	**81.5** ± 5.6	**54.0** ± 12.9	74.2 ± 16.5	60.4 ± 14.2	72.2 ± 2.1	76.1 ± 2.8	47.6 ± 5.9	56.8 ± 3.3
L2xGin	78.0 ± 0.6	49.5 ± 0.8	90.2 ± 1.4	63.8 ± 1.0	74.1 ± 4.3	65.6 ± 3.9	**82.8** ± 5.2	**70.7** ± 4.3
$L2xGin_{dsc}$	80.0 ± 1.2	52.1 ± 1.6	**94.7** ± 2.7	**67.1** ± 2.0	71.0 ± 3.0	62.4 ± 4.1	78.1 ± 3.2	66.9 ± 3.5

according to the validation accuracy and we compare it with the post-hoc techniques. In our case, we integrate L2xGnn into the same architecture and learn the edge masking during training as described before. In Table 5.4, we report the explanation accuracy evaluation with respect to the ground-truth motifs in comparison with post-hoc techniques. The explanation problem is formalized as a binary classification problem, where the edges belonging to the ground-truth motif are treated as positive labels. We observe that L2xGnn obtains better or the same results as the baseline GNN models. While for the post-hoc explanation techniques we cannot guarantee that the GNNs use exclusively the explanation subgraphs for the prediction [102], our method, by providing *faithful* explanations, overcomes this limitation. It is exactly the provided explanation that is used in the message-passing operations of L2xGnn.

Visual Evaluation of the Explanations. In Figure 5.2, we present some of the subgraphs identified by L2xGnn when combined with two different base GNNs. Based on prior studies and chemical domain knowledge [112], [115], [168], carbon rings (the blue circles in the pictures) and NO_2 groups are known to be mutagenic. Interestingly, we can notice that, when using the infor-

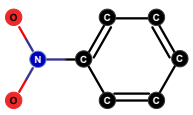


Figure 5.3: Benzene-NO_2 motif.

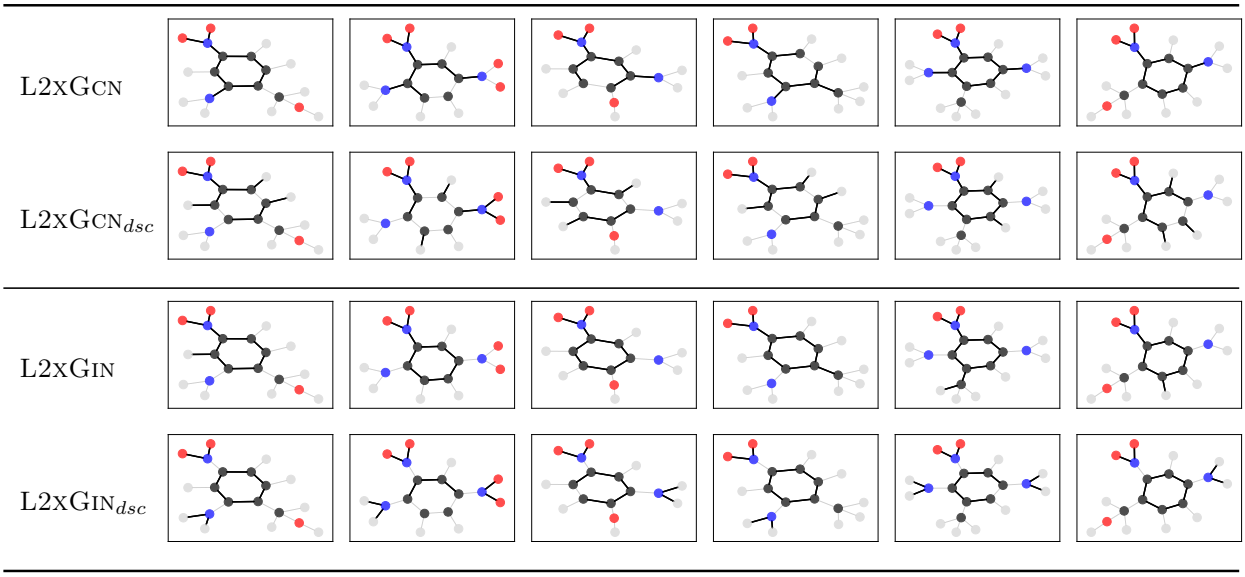


Figure 5.2: Visualization of some of the subgraphs selected by L2xGnn for $MUTAG_0$ on the test set. The solid edges represent the ones sampled by our approach. The subscript *dsc* indicates the maximum weight k-edge subgraph problem (i.e., possibly disconnected subgraphs). Black, blue, red, and gray nodes represent carbon (C), nitrogen (N), oxygen (O), and hydrogen (H) atoms respectively.

mation of connected subgraphs, the models are able to recognize a complete carbon ring with a NO_2 group in most of the cases. In some cases, the carbon ring is not complete, but the explanations are still helpful to understand which motifs are potentially important for the prediction. With the subscript *dsc*, we can observe the results of the sampling strategy when we do not require subgraphs to be connected. In this case, the carbon rings are not always identified. Instead, the NO_2 group is always considered important for the prediction. More generally, as also reported in Yuan, Yu, Wang, *et al.* [111], studying connected subgraphs results in more natural motifs compared to the motifs obtained without the connectedness constraint. A visual comparison of the explanations generated by L2xGnn and by the baselines can be found in Figure 5.6. The graph visualization supports the numerical evaluation carried on in Table 5.4. In fact, one can see that the explanations generated by the post-hoc approaches may vary substantially depending on the given input graph. In our case, instead, the explanations remain constant regardless of the input information. This claim supports the explanation accuracy analysis, where our approach has one of the smallest standard de-

viation among all the considered methods. Additionally, we also included the explanations generated with an attention-based GNN, namely GAT [21]. Although having a similar predictive performance in the graph classification task (99.6 ± 0.03), the resulting explanations are not qualitatively comparable with our approach. This is in line with previous works [106], [173], [175] asserting that graph attention models are not able to generate attention weights with high-fidelity, and consequently, cannot provide faithful and meaningful explanations.

Explanation Consistency. One crucial property for explanatory methods is consistency. For instance, if an explanation algorithm is applied to the same data instance multiple times, the generated explanations should be unchanged. Also, when different random seeds are used for the same architecture, the generated explanations should be stable. For the first case, we report the results in Table 5.5. Our method preserves its consistency when it is applied to the same data instance multiple times at test time. This is in line with the assumptions of our approach. In fact, since perturbations for subgraph sampling are removed at test time, this behavior is guaranteed. In Table 5.6, we report the explanation accuracy of our approach when using the same backbone architecture with different model initializations. Specifically, we compare a 3-layer GIN model using five different seeds for model initialization on the same data split. From the results, we can observe the ability of our method to generate consistent explanations irrespective of the differences across models.

Table 5.5: Explanation accuracy (%) on multiple test runs over the same data instances.

Dataset	BA-2MOTIFS			
	Acc.	Pr.	Rec.	F_1
L2XGIN	75.9 ± 0.0	47.0 ± 0.0	90.0 ± 0.0	61.7 ± 0.0
L2XGIN$_{dsc}$	77.9 ± 0.0	49.5 ± 0.0	94.4 ± 0.0	64.8 ± 0.0
Dataset	$MUTAG_0$			
	Acc.	Pr.	Rec.	F_1
L2XGIN	71.0 ± 0.0	63.7 ± 0.0	78.4 ± 0.0	67.7 ± 0.0
L2XGIN$_{dsc}$	70.8 ± 0.0	63.4 ± 0.0	77.0 ± 0.0	67.1 ± 0.0

Table 5.6: Explanation accuracy (%) on different model initializations using a 3-layer GIN architecture.

Dataset	BA-2MOTIFS			
	Acc.	Pr.	Rec.	F_1
L2xGin	75.9 ± 0.0	47.0 ± 0.0	90.0 ± 0.0	61.7 ± 0.0
L2xGin$_{dsc}$	77.9 ± 0.0	49.4 ± 0.0	94.3 ± 0.1	64.8 ± 0.0
Dataset	$MUTAG_0$			
	Acc.	Pr.	Rec.	F_1
L2xGin	73.8 ± 3.8	66.8 ± 3.7	81.8 ± 4.2	71.2 ± 3.8
L2xGin$_{dsc}$	68.7 ± 1.6	61.5 ± 2.6	74.4 ± 3.7	64.9 ± 1.9

Shortcut Learning Detection. By generating *faithful* subgraph explanations, our approach can be used to detect whether the predictive model is focusing on the expected features or if it is affected by shortcut learning. This is particularly important for GNNs, where seemingly small implementation differences can influence the learning process of the model [110]. To this end, we use the BA2Motifs dataset [107]. We trained two different models, GCN and GIN, achieving a test accuracy of 0.67 and 1.0 respectively. Taking a closer look at the explanations of the first model, we observed that most of the correct predictions were (incorrectly) correlated with the cycle motif and that the explanations were similar to the ones reported in Figure 5.4. The explanatory results show that the model is not learning the expected discriminative motifs and, consequently, the accuracy for the test set is poor. This insight can help users to change the configuration of the architecture or to use a different model (e.g., GIN). More generally, the results highlight that faithful explanations can facilitate model analysis and debugging.

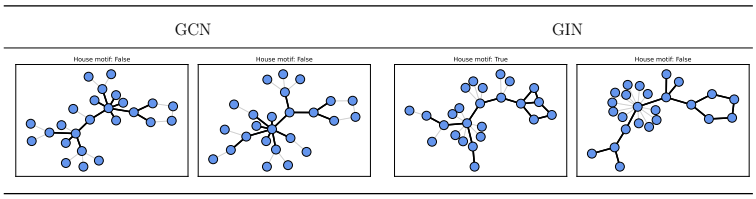


Figure 5.4: Example of model analysis based on the generated explanations.

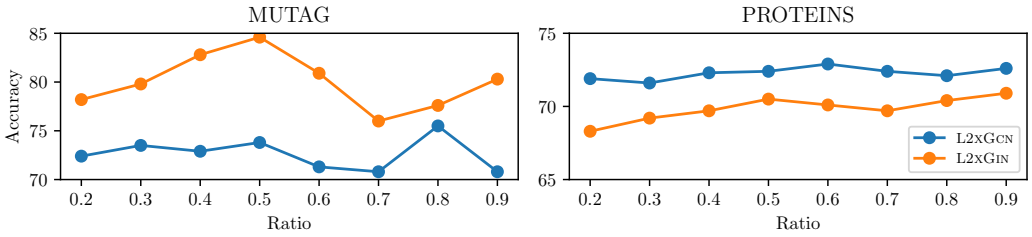


Figure 5.5: Effect of the edge ratio on the prediction accuracy (%).

Ablation Studies. In Table 5.3, we compare the two sampling strategies. From the results, the connected sampling is able to get better results than the non-connected counterpart on most datasets. In fact, the connectivity of subgraphs is essential to grasp the complete information about the important patterns, especially for chemical compound data where connected atoms are usually expected to create molecules or chemical groups. This aspect is also supported by the results obtained in the explanation accuracy task, where the connected strategy returns better explanations for the chemical dataset. Additionally, as previously mentioned, evaluating connected structures rather than just important edges looks more natural and intelligible. In Figure 5.5, we analyze the effect of the quantity of retained information on the prediction accuracy. A smaller ratio indicates that we retain fewer edges during training and, consequently, the resulting subgraphs are more sparse and, therefore, interpretable. As one can see, this affects the predictive capabilities only when R is small. Starting from $R = 0.5$, the ratio does not affect particularly the predictive capabilities of the model. In fact, for graph classification tasks, some of the information contained in the initial computational graph does not condition the prediction as the information may be redundant or noisy. For instance, considering the MUTAG dataset, we know that the initial graphs contain on average 20 edges. The discriminative motif benzene-NO_2, instead, contains around 9 edges, meaning that we ideally need 50% of the original edges to obtain good results. This is in line with the findings of this analysis and the graph classification results previously reported in Tables 5.3 and 5.4.

Comparison with Non-post-hoc Methods. Among the plethora of post-hoc methods for graphs, ProtGNN [125], KerGNN [176], and GSAT [173] are noteworthy exceptions. The first approach proposes a framework to generate explanations by comparing input graphs with prototypes learned during training, The second one combines graph kernels with the message passing paradigm to learn hidden graph filters. The latter, instead, leverages stochastic attention to select task-relevant subgraphs for interpretation. Although they all provide built-in explanations, given the introduction of new mechanisms to compute graph representations that differ from standard GNNs computations, the aforementioned approaches are not faithful by design (i.e., they do not reflect the reasoning process of the original backbone architecture). In contrast to these methods, our approach relies solely on standard GNNs, making it suitable to explain them faithfully. Additionally, in terms of explanatory capability, the learned prototypes are not directly interpretable and need to be matched to the closest training subgraphs to be human-understandable. Graph filters, instead, do not necessarily match existing patterns in the instance-based case. In both cases, the output can only provide a general idea of the important structures used by the model for prediction but fail at revealing precisely the instance-level explanation for each input graph.

Comparison with Graph Structure Learning Approaches. Recently, there have been related methods for learning the structure of graph neural networks. Following the taxonomy proposed in Zhu, Xu, Zhang, *et al.* [177], the structure learning methods most related to L2xGnn fall into the *postprocessing* category, and more specifically, under the *discrete sampling* subcategory. All existing methods use variants of the Gumbel-softmax trick which is limited in modeling complex distributions. Moreover, only when the straight-through version of the Gumbel-softmax trick is used, one can obtain truly discrete and not merely relaxed adjacency matrices in the forward pass. In contrast, L2xGnn always samples purely discrete adjacency matrices. It is, to the best of our knowledge, the only method that allows us to model complex dependencies between the edge variables through its ability to integrate

a combinatorial optimization algorithm on graphs. Other strategies include sampling edges between each pair of nodes from a Bernoulli distribution [178] or sampling subgraphs for subgraph aggregation methods in a data-driven manner [179]. All these methods, however, are not concerned with the problem of explaining the behavior of GNNs explicitly.

5.5 Summary

In this chapter, we propose L2xGnn, a framework that can be integrated into GNN architectures to learn to generate explanatory subgraphs which are exclusively used for the models' predictions. Our experimental findings demonstrate that the integration of L2xGnn with base GNNs does not affect the predictive capabilities of the model for graph classification tasks. Furthermore, according to the definition of faithfulness provided before, the resulting explanations are *faithful* since the retained information is the only one used by the model for prediction. Hence, differently from most of the common techniques, our explanations reveal the rationale of the GNNs and can also be used for model analysis and debugging.

A limitation of the approach is the reduced efficiency compared to baseline GNN models. Since we need to integrate an algorithm to compute (approximate) solutions to a combinatorial optimization problem, each forward-pass requires more time and resources. Moreover, depending on the choice of the optimization problem, we might not capture the structure of explanatory motifs required for the application under consideration. If for a medium graph two or more disconnected subgraphs are required, the algorithm `opt` would have to be changed to account for this.

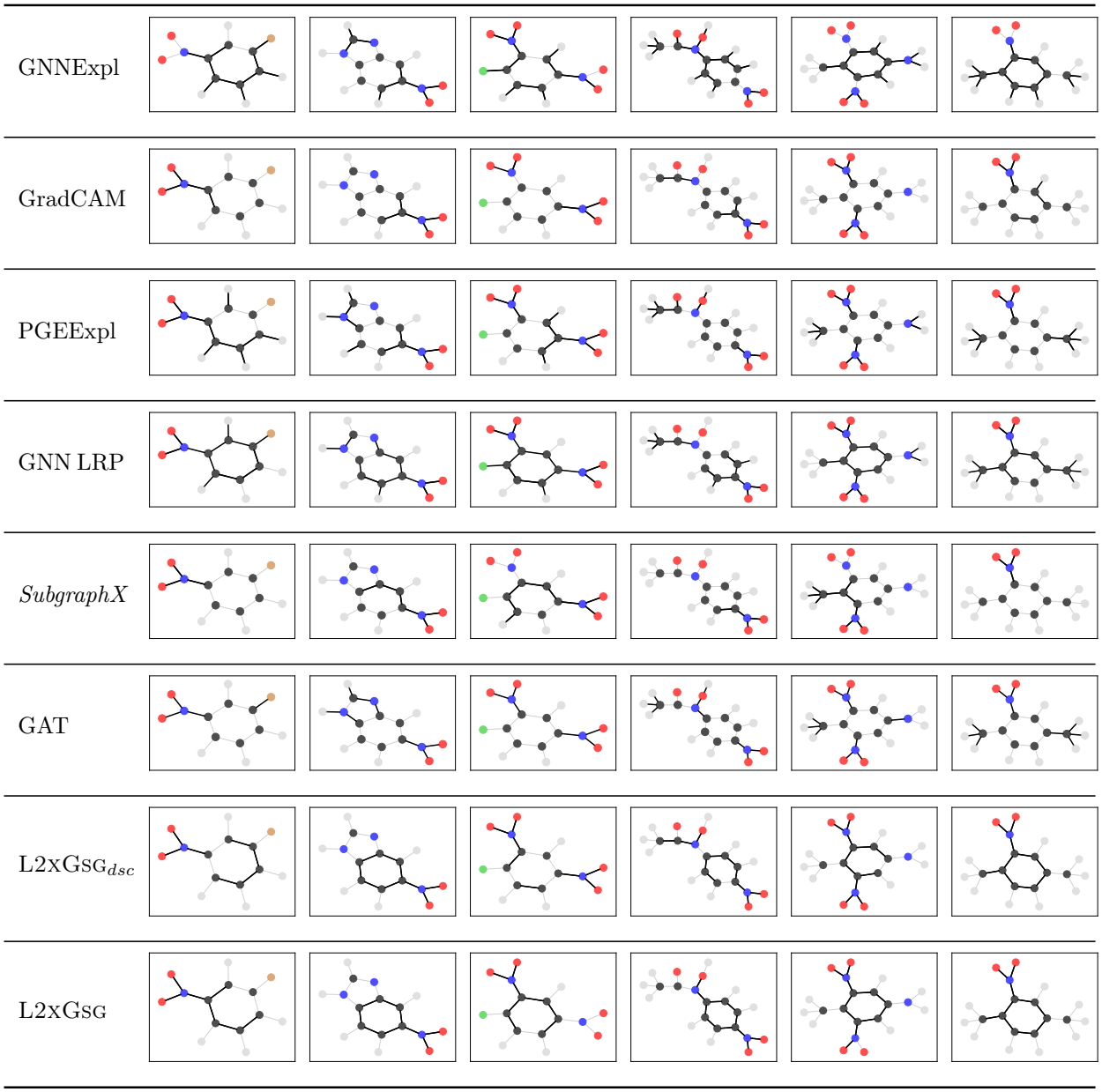


Figure 5.6: Comparison of the generated explanations for MUTAG$_0$ on the test set. The solid edges are the ones considered responsible of a correct prediction. Black, blue, red, gray, and green nodes represent carbon (C), nitrogen (N), oxygen (O), hydrogen (H), and chlorine (Cl) atoms respectively.

Chapter Six

Conclusions

With the increase of AI-based solutions in real-world applications comes a change in the requirements related to these methods. The wide adoption of machine learning models based on massive data information could create automatic decision systems that we cannot entirely understand. As a consequence, biased or unethical choices could occur more frequently. Thus, before exploiting decisions based upon black-box models, a series of concerns related to trustworthiness, faithfulness, and reliability should be taken into account. All of these aspects are related to what we now call Explainable AI (XAI).

Summary. Throughout this manuscript, we present novel approaches that could help humans better understand the reasoning behind a certain model-based decision. First, we propose alternative ways of creating the latent vector representations which are usually used by the models to encode the input information. In particular, using the additional textual information oftentimes available along with the original data, we generate interpretable vectors for which we exactly know the meaning of each dimension and value.

In Chapter 3, the creation of these interpretable text-based vectors is coupled with a learning task, namely, a rating prediction task. As a result, the learned vectors not only explain the characteristics of the instances but also reveal which information the model thinks is

relevant to the learning task. Additionally, by overcoming the limitation of common embedding techniques, the vectors can be evaluated and inspected both singularly and collectively. Indeed, through this method, we provide a new 'interactive' tool that can help understand the properties, differences, and similarities of the data considered. The evaluation of single vectors provides human-understandable information about specific instances (see Figure 3.4), while the evaluation of a group of vectors can give insights regarding the organization and properties of the whole dataset (Figure 3.2).

In Chapter 4, following a similar purpose, we present an alternative way of generating interpretable latent representations. Without relying on the neural design of common ML architectures, we propose a fully transparent probabilistic model that creates a topographic organization of latent classes. Differently from neural-based models where explanations are typically generated through a modification of a difficult-to-interpret network (e.g., by integrating an attention mechanism), in our case, interpretability is at the core of the proposed approach. Rather than implementing an architecture capable of creating explanations without a true understanding of the data, we present a probabilistic model grounded on data understanding which, in turn, is inherently interpretable. Also, in contrast with neural-based embedding techniques, instead of modeling each instance separately, we model classes of users and products. As a result, the latent codes are more compact and sparse. As before, the output of this probabilistic model lends itself to nice visualization and analysis of the results (Figure 4.3). Afterward, given the recent debates on the effective improvements of deep networks in comparison with '*old-fashioned*' approaches, we continue our analysis by using the learned vectors for a rating prediction task. The limited expressiveness of constrained low-dimensional representations, such as the ones presented here, should limit the ability of the learning model to discriminate the input information. However, the predictive results suggest otherwise. Although the simplicity of the information carried by our vectors, the results were similar compared to more sophisticated and high-dimensional representations. Surprisingly, also the combination of our interpretable compact vectors with

an interpretable model (i.e., a regression model), was able to achieve competitive results. Thus, the outcome of our experimental analysis suggests that, for some learning tasks, the use of complicated and deep architecture is not fully motivated. Instead, we could use approaches and representations much more interpretable without compromising the predictive capabilities significantly.

Finally, in Chapter 5, we take a step forward and present a novel approach that learns to explain the inner mechanisms of graph black-box models. Instead of trying to interpret the internal representations generated by the model, the main goal is to extract which part of the input is effectively used during inference for a given output (Figure 5.2). Our experimental findings suggest that the integration of L2xGnn with base GNNs, although filtering some information out, does not affect the predictive capabilities of the model for graph classification tasks. Furthermore, since we provide explanations that faithfully uncover the inner mechanisms of the base model, our approach can also be used to analyze potential learning problems (Figure 5.4) and modify the architecture to achieve the desired learning capabilities.

Evaluation of the Explanations. As previously stated, measuring the quality of explanations is still an open problem since a shared definition of what interpretability really is does not exist. In Chapter 3, in a textual context, we provide a quantitative evaluation of the generated explanations. The main motivation is that, instead of just relying on the visual human assessment, a metric could facilitate the immediate evaluation of the explanations. In our case, considering how textual information is processed by humans, we propose to use two different metrics. Starting from two different sets of words, we employ Jaccard similarity (see Eq. 3.12) to understand the direct correspondences between the words in the two sets, and the Word Mover's Distance (Eq. 3.13) to evaluate their semantic similarity. In this way, we can have a better idea of the quality of the explanations by both considering the lexical and semantic similarities of groups of words. Differently, in Chapter 5, we

make use of ground-truth datasets to evaluate the accuracy of our approach in detecting the discriminative motifs of the model's predictions. In the introductory chapters we mentioned that, usually, there might be some problems and discrepancies when evaluating explanatory methods with ground-truth datasets. This is particularly true when we analyze post-hoc explanations where we cannot assure a direct match between the generated explanations, the expected ones, and the information effectively used by the model. However, since our explanations faithfully reflect the model reasoning, the limitations related to the evaluation of post-hoc techniques no longer exist.

Future Works. To facilitate and guide the evaluation of the explanations, in Chapter 3, we propose two quantitative metrics for assessing the quality of the textual-based representations. In general, automatic-computed metrics can help humans evaluate the generated explanations rapidly. However, as already stated, interpretability remains something personal. One could find an explanation particularly insightful, while another one could find it meaningless or difficult to understand. Furthermore, depending on the data and objective, interpretability can have a different definition and, therefore, a universal metric cannot be created. For this reason, although it would be ideal to have machines evaluate automatically the explanations, we should make a step toward a more human-centric view of explainability and research in general. As we propose in [16], external stakeholders (i.e., research beneficiaries, end-users) should be involved by researchers in all research stages, from data collection to output evaluation. In this way, we would be able to fully understand the requirements and needs of the final 'evaluator', and we could provide models and outputs truly human-understandable.

Sometimes, apart from the predictive accuracy, it is important to understand the degree of confidence of the model for its predictions. The so-called *model uncertainty*, thanks to breakthrough applications as [180], has become popular and can help increase the transparency of the models. Through model uncertainty, we can provide a *measure of belief* about the

predictions. This is particularly significant in critical domains, where conventional point estimates could be insufficient for relying on a computer-based decision. In this scenario, to prevent wrong actions, confidence estimates would be beneficial to assist humans during the decision process. Similarly, also in the field of XAI, the introduction of model uncertainty estimates could increase human trust in ML-based models. In fact, in XAI as well, the generated explanations may be unstable and, therefore, uncertainty quantities could be useful to assess the reliability and stability of explanatory methods. The first attempt at modeling uncertainty for explainability was recently proposed in [181]. Motivated by the fact that small perturbations of the input could substantially change the corresponding explanation, the authors propose a Bayesian framework to generate explanations with uncertainty estimates for post-hoc scenarios. The direction is still new and deserves more attention in the future. For this reason, for future works, it would be interesting to develop new approaches that will be able to provide confidence measures along with the explanatory output.

To conclude, through the methodologies proposed in this thesis, we demonstrate that interpretable constrained models and representations may result in a relatively small reduction in terms of predictive capabilities. However, the gain in model and data understanding is so significant that we can sacrifice pure predictive performance to increase model transparency, especially for critical tasks. Following this direction, we also show that the use of deep and complicated systems is not fully motivated in some scenarios. Additionally, even though it would be convenient to have machines automatically evaluate the provided explanations, interpretability is still conditioned by the human perspective. Therefore, whether we want to understand the inner mechanisms of black-box models or their latent representations, we should move toward a human-centric view of research since only through a full interaction between humans and machines we can improve all the aspects related to XAI. We hope that the contributions and perspectives presented in this thesis can be a modest starting point for future evaluations on this matter.

Appendix One

Addendum to Chapter 3

In Chapter 3, we propose an approach to build text-based vector representations starting from a vocabulary of 2000 words. However, although the complexity and the high-dimensionality of the data, sparse representations are usually expected in many applications. For music or movie tag classification, for instance, among thousands of possible candidates, only a few of them are associated with each input sample. The same reasoning holds for the number of words associated with each user or product in an e-commerce system scenario. Despite the availability of thousands of words, a set of 20 words would probably be enough to understand the user's taste or product characteristics. Additionally, we may impose some sparsity constraints to limit the memory usage in limited-memory devices or simply to set a threshold on the maximum number of features to select. For all the above reasons, in [15], we present an approach to explicitly learn the sparsity of the latent representations with a twofold objective: a) model how sparse the representations should be; b) learn sparse data structures when a quantified sparsity constraint is in place. In this way, the learned sparsity degree is not fixed but fits the corresponding observation. Thanks to the flexibility in deciding the number of non-zero features to select while maintaining it below the defined threshold, the model reduces data dimensionality and is applicable when (hard) sparsity constraints are required. In order to achieve such flexibility, an auxiliary random variable is

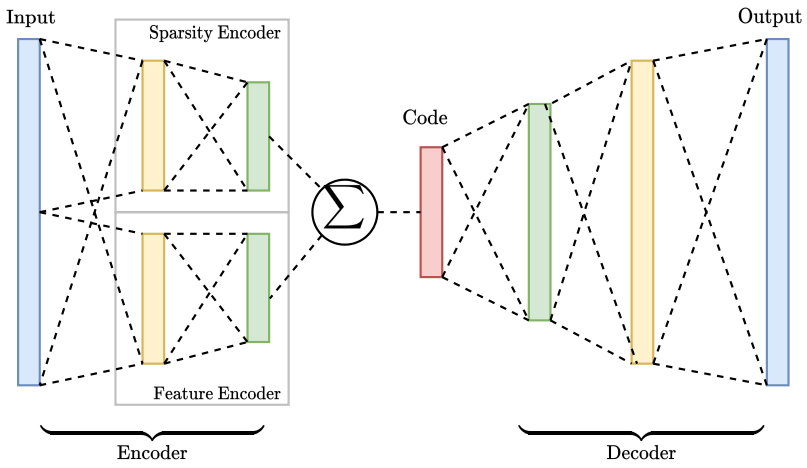


Figure A.1: Schematic view of the encoder-decoder architecture for learning sparse latent representations.

introduced to model the sparsity of the representations and to specify the maximum number of non-zero features required. More specifically, the model is a sparse deep latent generative model that combines the strength of deep learning approaches with the flexibility of Bayesian inference. In other words, we extend deep latent models [182]–[188] to explicitly model the sparsity of the representations. The inference is based on amortized variational methods [182], [183], [189]–[191], where we employ a encoder-decoder architecture for inference and learning. A schematic view of the architecture is presented in Figure A.1.

The creation of the sparse representations is based on a two-step sampling approach performed in the encoder part of the architecture. Basically, the encoder part consists of two different encoders: 1) a *feature encoder* which simply models the feature representation, as in a classical encoder; 2) a *sparsity encoder* to learn how sparse each latent vector should be. In this way, the model is capable of focusing on and extracting the important features only. The results of the two encoders are then combined through summation to create a single sparse vector representation which is in turn used in the decoder part to reconstruct the original input. We test the results in both unsupervised (image reconstruction) and supervised (multi-class multi-label classification) settings achieving in both cases competitive

and promising results.

Although we have not tried to integrate this idea in some of the approaches proposed in this thesis, this method may be potentially used to, e.g., further define the characteristics of the interpretable word-based vectors proposed in Chapter 3. In this context, we could set a limit on the number of words used to explain each item and let the architecture decide how many words are needed to generate a meaningful explanation for each user or product. In this way, by defining some additional properties of the explainable vectors, their interpretability can be further improved.

Appendix Two

Derivations of EM Algorithm Equations

B.1 E-step Derivations

First, we estimate $\hat{P}(\mathbf{z}_u^k, \mathbf{z}_p^\ell | w, u, p)$ as:

$$\frac{P(w|u,p,\mathbf{z}_u^k,\mathbf{z}_p^\ell)P(\mathbf{z}_u^k|u,p)P(\mathbf{z}_p^\ell|u,p)}{\sum_{k''}\sum_{\ell''}P(w|u,p,\mathbf{z}_u^{k''},\mathbf{z}_p^{\ell''})P(\mathbf{z}_u^{k''}|u,p)P(\mathbf{z}_p^{\ell''}|u,p)}. \tag{B.1}$$

By model assumptions, $\mathbf{z}_u$ is independent from products and $\mathbf{z}_p$ is independent from users, so we have $P(\mathbf{z}_u^k|u,p) = P(\mathbf{z}_u^k|u)$ and $P(\mathbf{z}_p^\ell|u,p) = P(\mathbf{z}_p^\ell|p)$. Consequently:

$$\begin{aligned} P(w|u,p,\mathbf{z}_u^k,\mathbf{z}_p^\ell) &= \sum_{k'}\sum_{\ell'} P(w,\mathbf{y}_u^{k'},\mathbf{y}_p^{\ell'}|u,p,\mathbf{z}_u^k,\mathbf{z}_p^\ell) \\ &= \sum_{k'}\sum_{\ell'}\left[P(w|u,p,\mathbf{z}_u^k,\mathbf{z}_p^\ell,\mathbf{y}_u^{k'},\mathbf{y}_p^{\ell'})P(\mathbf{y}_u^{k'}|u,p,\mathbf{z}_u^k)P(\mathbf{y}_p^{\ell'}|u,p,\mathbf{z}_p^\ell)\right] \\ &= \sum_{k'}\sum_{\ell'} P(w|\mathbf{y}_u^{k'},\mathbf{y}_p^{\ell'})P(\mathbf{y}_u^{k'}|\mathbf{z}_u^k)P(\mathbf{y}_p^{\ell'}|\mathbf{z}_p^\ell), \end{aligned} \tag{B.2}$$

and so $\hat{P}(\mathbf{z}_u^k, \mathbf{z}_p^\ell | w, u, p)$ is equal to

$$\frac{P(\mathbf{z}_u^k|u)P(\mathbf{z}_p^\ell|p)\sum_{k'}\sum_{\ell'}S(k,\ell)}{\sum_{k''}\sum_{\ell''}P(\mathbf{z}_u^k|u)P(\mathbf{z}_p^\ell|p)\sum_{k'}\sum_{\ell'}S(k'',\ell'')}, \tag{B.3}$$

with $S(\alpha,\beta) = P(w|\mathbf{y}_u^{k'},\mathbf{y}_p^{\ell'})P(\mathbf{y}_u^{k'}|\mathbf{z}_u^\alpha)P(\mathbf{y}_p^{\ell'}|\mathbf{z}_p^\beta)$.

Analogously, to compute $\hat{P}(\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'} | w, u, p)$, we have:

$$\frac{P(w|u,p,\mathbf{y}_u^{k'},\mathbf{y}_p^{\ell'})P(\mathbf{y}_u^{k'}|u,p)P(\mathbf{y}_p^{\ell'}|u,p)}{\sum_{k''}\sum_{\ell''}P(w|u,p,\mathbf{y}_u^{k''},\mathbf{y}_p^{\ell''})P(\mathbf{y}_u^{k''}|u,p)P(\mathbf{y}_p^{\ell''}|u,p)} \tag{B.4}$$

with

$$\begin{aligned} P(\mathbf{y}_u^{k'}|u,p) &= \sum_k P(\mathbf{y}_u^{k'}, \mathbf{z}_u^k|u,p) \\ &= \sum_k P(\mathbf{y}_u^{k'}|u,p,\mathbf{z}_u^k) P(\mathbf{z}_u^k|u,p) \\ &= \sum_k P(\mathbf{y}_u^{k'}|\mathbf{z}_u^k) P(\mathbf{z}_u^k|u). \end{aligned} \tag{B.5}$$

Following the same reasoning:

$$\begin{aligned} P(\mathbf{y}_p^{\ell'}|u,p) &= \sum_\ell P(\mathbf{y}_p^{\ell'}, \mathbf{z}_p^\ell|u,p) \\ &= \sum_\ell P(\mathbf{y}_p^{\ell'}|u,p,\mathbf{z}_p^\ell) P(\mathbf{z}_p^\ell|u,p) \\ &= \sum_\ell P(\mathbf{y}_p^{\ell'}|\mathbf{z}_p^\ell) P(\mathbf{z}_p^\ell|p). \end{aligned} \tag{B.6}$$

Hence, $\hat{P}(\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}|w,u,p)$ becomes:

$$\frac{P(w|\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}) \sum_k G(k') \sum_\ell W(\ell')}{\sum_{k''} \sum_{\ell''} P(w|\mathbf{y}_u^{k''}, \mathbf{y}_p^{\ell''}) \sum_k G(k'') \sum_\ell W(\ell'')}, \tag{B.7}$$

where $G(\alpha) = P(\mathbf{y}_u^\alpha|\mathbf{z}_u^k)P(\mathbf{z}_u^k|u)$ and $W(\beta) = P(\mathbf{y}_p^\beta|\mathbf{z}_p^\ell)P(\mathbf{z}_p^\ell|p)$.

B.2 M-step Derivations

For the M-step, we assume the functional forms of the distributions for $P(\mathbf{z}_u^k|u)$, $P(\mathbf{z}_p^\ell|p)$ and $P(w|\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'})$ as:

$$P(\mathbf{z}_u|u) \sim \text{Multinomial} \tag{B.8}$$

$$P(\mathbf{z}_p|p) \sim \text{Multinomial} \tag{B.9}$$

$$P(w|u,p) \sim \text{Multinomial}. \tag{B.10}$$

We introduce latent indicator variables $\delta_{k,k',\ell,\ell'}^{i,j}$ such that if the user u^i belonging to latent user class k (that was corrupted through channel noise to latent user class k') used the word

w_j^i when reviewing the product p^i belonging to latent product class ℓ (corrupted by channel noise to latent product class ℓ'), then $\delta_{k,k',\ell,\ell'}^{i,j} = 1$, otherwise $\delta_{k,k',\ell,\ell'}^{i,j} = 0$.

The complete-data log likelihood is then:

$$\begin{aligned}\mathcal{L}_C = &\sum_{i=1}^{R}\sum_{j=1}^{S_i}\sum_{k'=1}^{K}\sum_{\ell'=1}^{L}\sum_{k=1}^{K}\sum_{\ell=1}^{L}\delta_{k,k',\ell,\ell'}^{i,j} \\ &\Bigg[\log\Bigg(P(w_j^i|\mathbf{y}_{u^i}=k',\mathbf{y}_{p^i}=\ell') \\ &P(\mathbf{y}_{u^i}=k'|\mathbf{z}_{u^i}=k)P(\mathbf{z}_{u^i}=k|u^i) \\ &P(\mathbf{y}_{p^i}=\ell'|\mathbf{z}_{p^i}=\ell)P(\mathbf{z}_{p^i}=\ell|p^i)\Bigg)\Bigg].\end{aligned} \tag{B.11}$$

The expected complete data log-likelihood is:

$$\begin{aligned}\mathcal{L}_C = &\sum_{i=1}^{R}\sum_{j=1}^{S_i}\sum_{k'=1}^{K}\sum_{\ell'=1}^{L}\sum_{k=1}^{K}\sum_{\ell=1}^{L}E[\delta_{k,k',\ell,\ell'}^{i,j}] \\ &\Big[\log P(w_j^i|\mathbf{y}_{u^i}^{k'},\mathbf{y}_{p^i}^{\ell'}) + \log P(\mathbf{y}_{u^i}^{k'}|\mathbf{z}_{u^i}^{k}) \\ &+ \log P(\mathbf{y}_{p^i}^{\ell'}|\mathbf{z}_{p^i}^{\ell}) + \log P(\mathbf{z}_{u^i}^{k}|u^i) + \log P(\mathbf{z}_{p^i}^{\ell}|p^i)\Big].\end{aligned} \tag{B.12}$$

For convenience, we will write $\mathcal{L}_C$ as:

$$\begin{aligned}\mathcal{L}_C = &\sum_{i=1}^{R}\sum_{j=1}^{S_i}\Delta_{u,p,w}^{u^i,p^i,w_j^i}\sum_{k'=1}^{K}\sum_{\ell'=1}^{L}\sum_{k=1}^{K}\sum_{\ell=1}^{L}E[\delta_{k,k',\ell,\ell'}^{i,j}] \\ &\Big[\log P(w|\mathbf{y}_u^{k'},\mathbf{y}_p^{\ell'}) + \log P(\mathbf{y}_u^{k'}|\mathbf{z}_u^{k}) \\ &+ \log P(\mathbf{y}_p^{\ell'}|\mathbf{z}_p^{\ell}) + \log P(\mathbf{z}_u^{k}|u) + \log P(\mathbf{z}_p^{\ell}|p)\Big],\end{aligned} \tag{B.13}$$

where $\Delta_a^\alpha = 1$ if and only if $a = \alpha$, otherwise $\Delta_a^\alpha = 0$. We maximize the expected complete data log-likelihood extended with Lagrange multiplier terms:

$$\begin{aligned}
< \mathcal{L}_C > \quad = & \sum_{i=1}^{R} \sum_{j=1}^{S_i} \Delta_{u,p,w}^{u^i,p^i,w_j^i} \sum_{k'=1}^{K} \sum_{\ell'=1}^{L} \sum_{k=1}^{K} \sum_{\ell=1}^{L} \hat{P}(\mathbf{z}_u^k, \mathbf{z}_p^\ell, \mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'} | u, p, w) \\
& \Big[\log P(w | \mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}) + \log P(\mathbf{y}_u^{k'} | \mathbf{z}_u^k) \\
& + \log P(\mathbf{y}_p^{\ell'} | \mathbf{z}_p^\ell) + \log P(\mathbf{z}_u^k | u) + \log P(\mathbf{z}_p^\ell | p) \Big] \\
& + \sum_{u \in \mathcal{U}} \lambda_u \left(\sum_k P(\mathbf{z}_u^k | u) - 1 \right) \\
& + \sum_{p \in \mathcal{P}} \lambda_p \left(\sum_\ell P(\mathbf{z}_p^\ell | p) - 1 \right) \\
& + \sum_{k', \ell'} \lambda_{k', \ell'} \left(\sum_{w \in \mathcal{V}} P(w | \mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}) - 1 \right).
\end{aligned} \tag{B.14}$$

By setting $\frac{\partial < \mathcal{L}_C >}{\partial P(w | \mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'})} = 0$, we have:

$$\begin{aligned}
& \frac{\partial < \mathcal{L}_C >}{\partial P(w | \mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'})} = 0 \Longleftrightarrow \\
& \sum_{k=1}^{K} \sum_{\ell=1}^{L} \hat{P}(\mathbf{z}_u^k, \mathbf{z}_p^\ell, \mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'} | u, p, w) \frac{1}{P(w | \mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'})} + \lambda_{k', \ell'} = 0.
\end{aligned} \tag{B.15}$$

Then

$$\begin{aligned}
P(w | \mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}) &= - \frac{\sum\limits_{(u,p) \in \mathcal{B}_w} \sum\limits_k \sum\limits_\ell \hat{P}(\mathbf{z}_u^k, \mathbf{z}_p^\ell, \mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'} | u, p, w)}{\lambda_{k', \ell'}} \\
&= - \frac{\sum\limits_{(u,p) \in \mathcal{B}_w} \hat{P}(\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'} | u, p, w)}{\lambda_{k', \ell'}},
\end{aligned} \tag{B.16}$$

where $\mathcal{B}(w)$ is the set of (*user, product*) tuples associated with the word w. Substituting $P(w | \mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'})$ back to the constraint we have:

$$\begin{aligned}
& \sum_w P(w | \mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}) = 1 \Longleftrightarrow \\
& \lambda_{k', \ell'} = - \sum_w \sum_{(u,p) \in \mathcal{B}(w)} \hat{P}(\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'} | u, p, w),
\end{aligned} \tag{B.17}$$

and so

$$P(w|\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}) = \frac{\sum\limits_{(u,p)\in\mathcal{B}(w)} \hat{P}(\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}|u,p,w)}{\sum\limits_{w'}\sum\limits_{(u,p)\in\mathcal{B}(w')} \hat{P}(\mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}|u,p,w')}. \tag{B.18}$$

By setting $\frac{\partial<\mathcal{L}_C>}{\partial P(\mathbf{z}_u^k|u)} = 0$, we have:

$$\begin{aligned} &\frac{\partial < \mathcal{L}_C >}{\partial P(\mathbf{z}_u^k|u)} = 0 \Longleftrightarrow \\ &\sum_{k'=1}^{K}\sum_{\ell'=1}^{L}\sum_{\ell=1}^{L} \hat{P}(\mathbf{z}_u^k, \mathbf{z}_p^\ell, \mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}|u,p,w)\frac{1}{P(\mathbf{z}_u^k|u)} + \lambda_u = 0. \end{aligned} \tag{B.19}$$

Denoting the set of words used by user u to review product p by $\mathcal{W}(u,p)$, we obtain:

$$\begin{aligned} P(\mathbf{z}_u^k|u) &= -\frac{\sum\limits_{p}\sum\limits_{w\in\mathcal{W}(u,p)}\sum\limits_{k'}\sum\limits_{\ell'}\sum\limits_{\ell} \hat{P}(\mathbf{z}_u^k, \mathbf{z}_p^\ell, \mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}|u,p,w)}{\lambda_u} \\ &= -\frac{\sum\limits_{p}\sum\limits_{w\in\mathcal{W}(u,p)}\sum\limits_{\ell} \hat{P}(\mathbf{z}_u^k, \mathbf{z}_p^\ell|u,p,w)}{\lambda_u}. \end{aligned} \tag{B.20}$$

Substituting $P(\mathbf{z}_u^k|u)$ back to the constraint we have:

$$\begin{aligned} &\sum_{k=1}^{K} P(\mathbf{z}_u^k|u) = 1 \\ &\Longleftrightarrow \lambda_u = -\sum_{k}\sum_{\ell}\sum_{p}\sum_{w\in\mathcal{W}(u,p)} \hat{P}(\mathbf{z}_u^k, \mathbf{z}_p^\ell|u,p,w) \\ &\Longleftrightarrow \lambda_u = -\sum_{p}|\mathcal{W}(u,p)| \end{aligned} \tag{B.21}$$

and so

$$P(\mathbf{z}_u^k|u) = \frac{\sum_p\sum_{w\in\mathcal{W}(u,p)}\sum_\ell \hat{P}(\mathbf{z}_u^k, \mathbf{z}_p^\ell|u,p,w)}{\sum_p|\mathcal{W}(u,p)|}. \tag{B.22}$$

By setting $\frac{\partial<\mathcal{L}_C>}{\partial P(\mathbf{z}_p^\ell|p)} = 0$, we have:

$$\begin{aligned} &\frac{\partial < \mathcal{L}_C >}{\partial P(\mathbf{z}_p^\ell|p)} = 0 \Longleftrightarrow \\ &\sum_{k'=1}^{K}\sum_{\ell'=1}^{L}\sum_{k=1}^{K} \hat{P}(\mathbf{z}_u^k, \mathbf{z}_p^\ell, \mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}|u,p,w)\frac{1}{P(\mathbf{z}_p^\ell|p)} + \lambda_p = 0. \end{aligned} \tag{B.23}$$

Then,

$$\begin{aligned} P(\mathbf{z}_p^\ell|p) &= -\frac{\sum_u \sum_{w\in\mathcal{W}(u,p)} \sum_{k'} \sum_{\ell'} \sum_k \hat{P}(\mathbf{z}_u^k, \mathbf{z}_p^\ell, \mathbf{y}_u^{k'}, \mathbf{y}_p^{\ell'}|u,p,w)}{\lambda_p} \\ &= -\frac{\sum_u \sum_{w\in\mathcal{W}(u,p)} \sum_k \hat{P}(\mathbf{z}_u^k, \mathbf{z}_p^\ell|u,p,w)}{\lambda_p}. \end{aligned} \tag{B.24}$$

Substituting $P(\mathbf{z}_p^\ell|p)$ back to the constraint we have:

$$\begin{aligned} &\sum_\ell P(\mathbf{z}_p^\ell|p) = 1 \\ &\iff \lambda_p = -\sum_k \sum_\ell \sum_u \sum_{w\in\mathcal{W}(u,p)} \hat{P}(\mathbf{z}_u^k, \mathbf{z}_p^\ell|u,p,w) \\ &\iff \lambda_p = -\sum_u |\mathcal{W}(u,p)| \end{aligned} \tag{B.25}$$

and finally:

$$P(\mathbf{z}_p^\ell|p) = \frac{\sum_u \sum_{w\in\mathcal{W}(u,p)} \sum_k \hat{P}(\mathbf{z}_u^k, \mathbf{z}_p^\ell|u,p,w)}{\sum_u |\mathcal{W}(u,p)|}. \tag{B.26}$$

References

[1] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-lm: Training multi-billion parameter language models using model parallelism," *arXiv preprint arXiv:1909.08053*, 2019.

[2] T. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[3] D. Lepikhin, H. Lee, Y. Xu, *et al.*, "{gs}hard: Scaling giant models with conditional computation and automatic sharding," in *International Conference on Learning Representations*, 2021.

[4] P. Goyal, M. Caron, B. Lefaudeux, *et al.*, "Self-supervised pretraining of visual features in the wild," *arXiv preprint arXiv:2103.01988*, 2021.

[5] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[6] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artifical Intelligence and Machine Learning*, vol. 14, no. 3, pp. 1–159, 2020.

[7] P. D. Hoff, A. E. Raftery, and M. S. Handcock, "Latent space approaches to social network analysis," *Journal of the american Statistical association*, vol. 97, no. 460, pp. 1090–1098, 2002.

[8] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[9] J. Lin, "The neural hype and comparisons against weak baselines," in *ACM SIGIR Forum*, ACM New York, NY, USA, vol. 52, 2019, pp. 40–51.

[10] W. Yang, K. Lu, P. Yang, and J. Lin, "Critically examining the" neural hype" weak baselines and the additivity of effectiveness gains from neural ranking models," in *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, 2019, pp. 1129–1132.

[11] R. Geirhos, J.-H. Jacobsen, C. Michaelis, *et al.*, "Shortcut learning in deep neural networks," *Nature Machine Intelligence*, vol. 2, no. 11, pp. 665–673, 2020.

[12] J. Chen, L. Song, M. Wainwright, and M. Jordan, "Learning to explain: An information-theoretic perspective on model interpretation," in *International Conference on Machine Learning*, PMLR, 2018, pp. 883–892.

[13] G. Serra, Z. Xu, M. Niepert, C. Lawrence, P. Tiňo, and X. Yao, "Interpreting node embedding with text-labeled graphs," in *2021 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2021, pp. 1–8.

[14] G. Serra and M. Niepert, "Learning to explain graph neural networks," *arXiv preprint arXiv:2209.14402*, 2022.

[15] Z. Xu, D. O. Rubio, G. Serra, and M. Niepert, "Learning sparsity of representations with discrete latent variables," in *2021 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2021, pp. 1–9.

[16] B. Kotnis, K. Gashteovski, J. Gastinger, *et al.*, "Human-centric research for nlp: Towards a definition and guiding questions," in *HCI+NLP Workshop at 2022 Annual*

Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), 2022.

[17] R. Wilson, *Introduction to Graph Theory*. Longman, 1996.

[18] R. Diestel, *Graph Theory*. Springer Berlin Heidelberg, 2016.

[19] P. Veličković, "Message passing all the way up," in *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022.

[20] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 1025–1035.

[21] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[22] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" In *International Conference on Learning Representations*, 2018.

[23] A. Duval and F. D. Malliaros, "Graphsvx: Shapley value explanations for graph neural networks," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2021, pp. 302–318.

[24] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," *Advances in neural information processing systems*, vol. 31, 2018.

[25] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[26] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *International conference on machine learning*, PMLR, 2019, pp. 3734–3743.

[27] E. Luzhnica, B. Day, and P. Lio, "Clique pooling for graph classification," *arXiv preprint arXiv:1904.00374*, 2019.

[28] J. Huang, Z. Li, N. Li, S. Liu, and G. Li, "Attpool: Towards hierarchical feature representation in graph convolutional networks via attention mechanism," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6480–6489.

[29] Y. Ma, S. Wang, C. C. Aggarwal, and J. Tang, "Graph convolutional networks with eigenpooling," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 723–731.

[30] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[31] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3693–3702.

[32] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*, PMLR, 2016, pp. 2014–2023.

[33] H. Xuanyuan, P. Barbiero, D. Georgiev, L. C. Magister, and P. Lió, "Global concept-based interpretability for graph neural networks via neuron analysis," *arXiv preprint arXiv:2208.10609*, 2022.

[34] G. Dall'Aglio, *Calcolo delle Probabilità*. Zanichelli, 2003.

[35] G. Casella and R. L. Berger, *Statistical inference*. Cengage Learning, 2021.

[36] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society: series B (methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[37] I. Lage, E. Chen, J. He, *et al.*, "An evaluation of the human-interpretability of explanation," *arXiv preprint arXiv:1902.00006*, 2019.

[38] H. Lakkaraju, D. Slack, Y. Chen, C. Tan, and S. Singh, "Rethinking explainability as a dialogue: A practitioner's perspective," *arXiv preprint arXiv:2202.01875*, 2022.

[39] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.

[40] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information fusion*, vol. 58, pp. 82–115, 2020.

[41] C. Molnar, *Interpretable Machine Learning, A Guide for Making Black Box Models Explainable*, 2nd ed. 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book.

[42] G. Ras, N. Xie, M. van Gerven, and D. Doran, "Explainable deep learning: A field guide for the uninitiated," *Journal of Artificial Intelligence Research*, vol. 73, pp. 329–397, 2022.

[43] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong, "Interpretable machine learning: Fundamental principles and 10 grand challenges," *Statistic Surveys*, vol. 16, pp. 1–85, 2022.

[44] P. Schmidt and F. Biessmann, "Quantifying interpretability and trust in machine learning systems," *arXiv preprint arXiv:1901.08558*, 2019.

[45] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[46] L. Faber, A. K. Moghaddam, and R. Wattenhofer, "When comparing to ground truth is wrong: On evaluating gnn explanation methods," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 332–341.

[47] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[48] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[49] M. T. Ribeiro, S. Singh, and C. Guestrin, "Nothing else matters: Model-agnostic explanations by identifying prediction invariance," *arXiv preprint arXiv:1611.05817*, 2016.

[50] X. Zhao, W. Huang, X. Huang, V. Robu, and D. Flynn, "Baylime: Bayesian local interpretable model-agnostic explanations," in *Uncertainty in Artificial Intelligence*, PMLR, 2021, pp. 887–896.

[51] A. Perotti, P. Bajardi, F. Bonchi, and A. Panisson, "Graphshap: Motif-based explanations for black-box graph classifiers," *arXiv preprint arXiv:2202.08815*, 2022.

[52] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[53] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *International conference on machine learning*, PMLR, 2017, pp. 3319–3328.

[54] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[55] C. Lawrence, T. Sztyler, and M. Niepert, "Explaining neural matrix factorization with gradient rollback.," in *AAAI*, 2021, pp. 4987–4995.

[56] F. Wang and C. Rudin, "Falling rule lists," in *Artificial intelligence and statistics*, PMLR, 2015, pp. 1013–1022.

[57] H. Lakkaraju, S. H. Bach, and J. Leskovec, "Interpretable decision sets: A joint framework for description and prediction," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1675–1684.

[58] N. Elhage, N. Nanda, C. Olsson, *et al.*, "A mathematical framework for transformer circuits," *Transformer Circuits Thread*, vol. 1, p. 1, 2021.

[59] W. Gurnee and M. Tegmark, "Language models represent space and time," in *The Twelfth International Conference on Learning Representations*, 2023.

[60] A. Chattopadhyay, S. Slocum, B. D. Haeffele, R. Vidal, and D. Geman, "Interpretable by design: Learning predictors by composing interpretable queries," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 6, pp. 7430–7443, 2022.

[61] T. Speith, "A review of taxonomies of explainable artificial intelligence (xai) methods," in *2022 ACM Conference on Fairness, Accountability, and Transparency*, 2022, pp. 2239–2250.

[62] W. Samek and K.-R. Müller, "Towards explainable artificial intelligence," in *Explainable AI: interpreting, explaining and visualizing deep learning*, Springer, 2019, pp. 5–22.

[63] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[64] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[65] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," *Advances in neural information processing systems*, vol. 14, 2001.

[66] S. Zhang, H. Tong, J. Xu, and R. Maciejewski, "Graph convolutional networks: A comprehensive review," *Computational Social Networks*, vol. 6, no. 1, pp. 1–23, 2019.

[67] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[68] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*, 2015, pp. 1067–1077.

[69] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[70] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec," in *Proceedings of the eleventh ACM international conference on web search and data mining*, 2018, pp. 459–467.

[71] C. Hacker, "k-simplex2vec: A simplicial extension of node2vec," in *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*, 2020.

[72] Y. Wang, L. Dong, X. Jiang, X. Ma, Y. Li, and H. Zhang, "Kg2vec: A node2vec-based vectorization model for knowledge graph," *Plos one*, vol. 16, no. 3, e0248552, 2021.

[73] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, 2013.

[74] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[75] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," *Advances in neural information processing systems*, vol. 26, 2013.

[76] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *International Conference on Learning Representations*, 2018.

[77] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 974–983.

[78] C. Tu, Z. Zhang, Z. Liu, and M. Sun, "Transnet: Translation-based network representation learning for social relation extraction.," in *IJCAI*, 2017, pp. 2864–2870.

[79] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Chang, "Network representation learning with rich text information," in *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[80] D. Shen, X. Zhang, R. Henao, and L. Carin, "Improved semantic-aware network embedding with fine-grained word alignment," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 1829–1838.

[81] S. Al-Sayouri, E. Gujral, D. Koutra, E. E. Papalexakis, and S. S. Lam, "T-pine: Tensor-based predictable and interpretable node embeddings," *Social Network Analysis and Mining*, vol. 10, no. 1, pp. 1–11, 2020.

[82] C. T. Duong, Q. V. H. Nguyen, and K. Aberer, "Interpretable node embeddings with mincut loss," in *Learning and Reasoning with Graph-Structured Representations Workshop-ICML*, 2019.

[83] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[84] C. M. Bishop, M. Svensén, and C. K. Williams, "Gtm: The generative topographic mapping," *Neural computation*, vol. 10, no. 1, pp. 215–234, 1998.

[85] T. Hofmann, "Probmap–a probabilistic approach for mapping large document collections," *Intelligent Data Analysis*, vol. 4, no. 2, pp. 149–164, 2000.

[86] G. Polčicová and P. Tiňo, "Making sense of sparse rating data in collaborative filtering via topographic organization of user preference patterns," *Neural Networks*, vol. 17, no. 8-9, pp. 1183–1199, 2004.

[87] S. Seo, J. Huang, H. Yang, and Y. Liu, "Representation learning of users and items for review rating prediction using attention-based convolutional neural network," in *International Workshop on Machine Learning Methods for Recommender Systems*, 2017.

[88] N. Jakob, S. H. Weber, M. C. Müller, and I. Gurevych, "Beyond the stars: Exploiting free-text user reviews to improve the accuracy of movie recommendations," in *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, 2009, pp. 57–64.

[89] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: Understanding rating dimensions with review text," in *Proceedings of the 7th ACM conference on Recommender systems*, 2013, pp. 165–172.

[90] G. Ling, M. R. Lyu, and I. King, "Ratings meet reviews, a combined approach to recommend," in *Proceedings of the 8th ACM Conference on Recommender systems*, 2014, pp. 105–112.

[91] Y. Bao, H. Fang, and J. Zhang, "Topicmf: Simultaneously exploiting ratings and reviews for recommendation," in *Twenty-Eighth AAAI conference on artificial intelligence*, 2014.

[92] A. Almahairi, K. Kastner, K. Cho, and A. Courville, "Learning distributed representations from reviews for collaborative filtering," in *Proceedings of the 9th ACM Conference on Recommender Systems*, 2015, pp. 147–154.

[93] A. Garcia-Duran, R. Gonzalez, D. Onoro-Rubio, M. Niepert, and H. Li, "Transrev: Modeling reviews as translations from users to items," in *European Conference on Information Retrieval*, Springer, 2020, pp. 234–248.

[94] Y. Zhang, X. Chen, *et al.*, "Explainable recommendation: A survey and new perspectives," *Foundations and Trends® in Information Retrieval*, vol. 14, no. 1, pp. 1–101, 2020.

[95] M. F. Dacrema, P. Cremonesi, and D. Jannach, "Are we really making much progress? a worrying analysis of recent neural recommendation approaches," in *Proceedings of the 13th ACM conference on recommender systems*, 2019, pp. 101–109.

[96] Y. Xian, Z. Fu, H. Zhao, *et al.*, "Cafe: Coarse-to-fine neural symbolic reasoning for explainable recommendation," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 1645–1654.

[97] C. Chen, M. Zhang, Y. Liu, and S. Ma, "Neural attentional rating regression with review-level explanations," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 1583–1592.

[98] S. Yu, Y. Wang, M. Yang, B. Li, Q. Qu, and J. Shen, "Nairs: A neural attentive interpretable recommendation system," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019, pp. 790–793.

[99] X. Dong, J. Ni, W. Cheng, *et al.*, "Asymmetrical hierarchical networks with attentive interactions for interpretable review-based recommendation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 7667–7674.

[100] L. Hu, S. Jian, L. Cao, and Q. Chen, "Interpretable recommendation via attraction modeling: Learning multilevel attractiveness over multimodal movie contents," in *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

[101] T. Chen, H. Yin, G. Ye, Z. Huang, Y. Wang, and M. Wang, "Try this instead: Personalized and interpretable substitute recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 891–900.

[102] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[103] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, and H. Hoffmann, "Explainability methods for graph convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 772–10 781.

[104] F. Baldassarre and H. Azizpour, "Explainability techniques for graph convolutional networks," *arXiv preprint arXiv:1905.13686*, 2019.

[105] B. Sanchez-Lengeling, J. Wei, B. Lee, *et al.*, "Evaluating attribution for graph neural networks," *Advances in neural information processing systems*, vol. 33, pp. 5898–5910, 2020.

[106] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," *Advances in neural information processing systems*, vol. 32, p. 9240, 2019.

[107] D. Luo, W. Cheng, D. Xu, *et al.*, "Parameterized explainer for graph neural network," *Advances in neural information processing systems*, vol. 33, pp. 19 620–19 631, 2020.

[108] T. Funke, M. Khosla, M. Rathee, and A. Anand, "Z orro: Valid, sparse, and stable explanations in graph neural networks," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[109] D. Loveland, S. Liu, B. Kailkhura, A. Hiszpanski, and Y. Han, "Reliable graph neural network explanations through adversarial training," *arXiv preprint arXiv:2106.13427*, 2021.

[110] M. S. Schlichtkrull, N. D. Cao, and I. Titov, "Interpreting graph neural networks for {nlp} with differentiable edge masking," in *International Conference on Learning Representations*, 2021.

[111] H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji, "On explainability of graph neural networks via subgraph explorations," in *International Conference on Machine Learning*, PMLR, 2021, pp. 12 241–12 252.

[112] W. Lin, H. Lan, and B. Li, "Generative causal explanations for graph neural networks," in *International Conference on Machine Learning*, PMLR, 2021, pp. 6666–6679.

[113] W. Lin, H. Lan, H. Wang, and B. Li, "Orphicx: A causality-inspired latent variable model for interpreting graph neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 729–13 738.

[114] A. Lucic, M. A. Ter Hoeve, G. Tolomei, M. De Rijke, and F. Silvestri, "Cf-gnnexplainer: Counterfactual explanations for graph neural networks," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2022, pp. 4499–4511.

[115] J. Tan, S. Geng, Z. Fu, *et al.*, "Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1018–1027.

[116] Q. Huang, M. Yamada, Y. Tian, D. Singh, and Y. Chang, "Graphlime: Local interpretable model explanations for graph neural networks," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[117] M. Vu and M. T. Thai, "Pgm-explainer: Probabilistic graphical model explanations for graph neural networks," *Advances in neural information processing systems*, vol. 33, pp. 12 225–12 235, 2020.

[118] Y. Zhang, D. Defazio, and A. Ramesh, "Relex: A model-agnostic relational model explainer," in *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, 2021, pp. 1042–1049.

[119] S. Gui, H. Yuan, J. Wang, Q. Lao, K. Li, and S. Ji, "Flowx: Towards explainable graph neural networks via message flows," *arXiv preprint arXiv:2206.12987*, 2022.

[120] R. Schwarzenberg, M. Hübner, D. Harbecke, C. Alt, and L. Hennig, "Layerwise relevance visualization in convolutional text graph classifiers," in *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (Text Graphs-13)*, 2019, pp. 58–62.

[121] J. Hu, T. Li, and S. Dong, "Gcn-lrp explanation: Exploring latent attention of graph convolutional networks," in *2020 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2020, pp. 1–8.

[122] T. Schnake, O. Eberle, J. Lederer, *et al.*, "Higher-order explanations of graph neural networks via relevant walks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 11, pp. 7581–7596, 2021.

[123] Q. Feng, N. Liu, F. Yang, R. Tang, M. Du, and X. Hu, "DEGREE: Decomposition based explanation for graph neural networks," in *International Conference on Learning Representations*, 2022.

[124] H. Yuan, J. Tang, X. Hu, and S. Ji, "Xgnn: Towards model-level explanations of graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 430–438.

[125] Z. Zhang, Q. Liu, H. Wang, C. Lu, and C. Lee, "Protgnn: Towards self-explaining graph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 9127–9135.

[126] L. C. Magister, D. Kazhdan, V. Singh, and P. Liò, "Gcexplainer: Human-in-the-loop concept-based explanations for graph neural networks," *arXiv preprint arXiv:2107.11889*, 2021.

[127] Y. Gao, T. Sun, R. Bhatt, D. Yu, S. Hong, and L. Zhao, "Gnes: Learning to explain graph neural networks," in *2021 IEEE International Conference on Data Mining (ICDM)*, IEEE, 2021, pp. 131–140.

[128] Z. Yu and H. Gao, "Motifexplainer: A motif-based graph neural network explainer," *arXiv preprint arXiv:2202.00519*, 2022.

[129] C. Agarwal, M. Zitnik, and H. Lakkaraju, "Probing gnn explainers: A rigorous theoretical and empirical analysis of gnn explanation methods," in *International Conference on Artificial Intelligence and Statistics*, 2022, pp. 8969–8996.

[130] C. Agarwal, O. Queen, H. Lakkaraju, and M. Zitnik, "An explainable ai library for benchmarking graph explainers," 2022.

[131] T. Hamaguchi, H. Oiwa, M. Shimbo, and Y. Matsumoto, "Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 1802–1808.

[132] S. Rhee, S. Seo, and S. Kim, "Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification," in *IJCAI*, 2018.

[133] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, "Protein interface prediction using graph convolutional networks," *Advances in neural information processing systems*, vol. 30, 2017.

[134] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an, "Graph convolutional encoders for syntax-aware neural machine translation," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1957–1967.

[135] D. Beck, G. Haffari, and T. Cohn, "Graph-to-sequence learning using gated graph neural networks," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 273–283.

[136] L. Song, Y. Zhang, Z. Wang, and D. Gildea, "N-ary relation extraction using graph-state lstm," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2226–2235.

[137] Y. Zhang, P. Qi, and C. D. Manning, "Graph convolution over pruned dependency trees improves relation extraction," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2205–2215.

[138] M. Narasimhan, S. Lazebnik, and A. Schwing, "Out of the box: Reasoning with graph convolution nets for factual visual question answering," *Advances in neural information processing systems*, vol. 31, 2018.

[139] D. Teney, L. Liu, and A. van Den Hengel, "Graph-structured representations for visual question answering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1–9.

[140] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, "Relation networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3588–3597.

[141] Y. Miao, L. Yu, and P. Blunsom, "Neural variational inference for text processing," in *International conference on machine learning*, PMLR, 2016, pp. 1727–1736.

[142] Y. Miao, E. Grefenstette, and P. Blunsom, "Discovering discrete latent topics with neural variational inference," in *International conference on machine learning*, PMLR, 2017, pp. 2410–2419.

[143] A. Martins and R. Astudillo, "From softmax to sparsemax: A sparse model of attention and multi-label classification," in *International conference on machine learning*, PMLR, 2016, pp. 1614–1623.

[144] A. Laha, S. A. Chemmengath, P. Agrawal, M. Khapra, K. Sankaranarayanan, and H. G. Ramaswamy, "On controllable sparse alternatives to softmax," *Advances in neural information processing systems*, vol. 31, 2018.

[145] X. Yan, J. Guo, Y. Lan, and X. Cheng, "A biterm topic model for short texts," in *Proceedings of the 22nd international conference on World Wide Web*, 2013, pp. 1445–1456.

[146] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[147] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," in *Proceedings of the tenth ACM international conference on web search and data mining*, 2017, pp. 425–434.

[148] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[149] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger, "From word embeddings to document distances," in *International conference on machine learning*, PMLR, 2015, pp. 957–966.

[150] X. Zhao and G. Serra, *Interpretable node embedding*, US Patent App. 16/841,762, Oct. 2021.

[151] G. Lample, M. Ott, A. Conneau, L. Denoyer, and M. Ranzato, "Phrase-based & neural unsupervised machine translation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 5039–5049.

[152] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[153] N. Parmar, A. Vaswani, J. Uszkoreit, *et al.*, "Image transformer," in *International Conference on Machine Learning*, PMLR, 2018, pp. 4055–4064.

[154] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2020.

[155] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 1419–1428.

[156] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 346–353.

[157] M. F. Dacrema, S. Boglio, P. Cremonesi, and D. Jannach, "A troubling analysis of reproducibility and progress in recommender systems research," *ACM Transactions on Information Systems (TOIS)*, vol. 39, no. 2, pp. 1–49, 2021.

[158] M. Ludewig and D. Jannach, "Evaluation of session-based recommendation algorithms," *User Modeling and User-Adapted Interaction*, vol. 28, no. 4, pp. 331–390, 2018.

[159] D. Jannach and M. Ludewig, "When recurrent neural networks meet the neighborhood for session-based recommendation," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, 2017, pp. 306–310.

[160] Y. Tay, A. T. Luu, and S. C. Hui, "Multi-pointer co-attention networks for recommendation," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2309–2318.

[161] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[162] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[163] L. K. Şenel, I. Utlu, F. Şahinuç, H. M. Ozaktas, and A. Koç, "Imparting interpretability to word embeddings while preserving semantic structure," *Natural Language Engineering*, vol. 27, no. 6, pp. 721–746, 2021.

[164] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. manuscript based on c. rudin please stop explaining black box machine learning models for high stakes decisions," in *Proceed-*

ings of NeurIPS 2018 Workshop on Critiquing and Correcting Trends in Learning, 2018.

[165] M. Niepert, P. Minervini, and L. Franceschi, "Implicit mle: Backpropagating through discrete exponential family distributions," *Advances in Neural Information Processing Systems*, vol. 34, pp. 14 567–14 579, 2021.

[166] G. Papandreou and A. L. Yuille, "Perturb-and-map random fields: Using discrete optimization to learn and sample from energy models," in *2011 International Conference on Computer Vision*, 2011, pp. 193–200.

[167] J. Domke, "Implicit differentiation by perturbation," *Advances in Neural Information Processing Systems*, vol. 23, pp. 523–531, 2010.

[168] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity," *Journal of medicinal chemistry*, vol. 34, no. 2, pp. 786–797, 1991.

[169] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. Vishwanathan, A. J. Smola, and H.-P. Kriegel, "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, no. suppl_1, pp. i47–i56, 2005.

[170] X. Yan, H. Cheng, J. Han, and P. S. Yu, "Mining significant graph patterns by leap search," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 433–444.

[171] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1365–1374.

[172] R. Rossi and N. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[173] S. Miao, M. Liu, and P. Li, "Interpretable and generalizable graph learning via stochastic attention mechanism," in *International Conference on Machine Learning*, PMLR, 2022, pp. 15 524–15 543.

[174] M. Liu, Y. Luo, L. Wang, *et al.*, "DIG: A turnkey library for diving into graph deep learning research," *Journal of Machine Learning Research*, vol. 22, no. 240, pp. 1–9, 2021. [Online]. Available: http://jmlr.org/papers/v22/21-0343.html.

[175] J. Yu, T. Xu, Y. Rong, Y. Bian, J. Huang, and R. He, "Graph information bottleneck for subgraph recognition," in *International Conference on Learning Representations*, 2020.

[176] A. Feng, C. You, S. Wang, and L. Tassiulas, "Kergnns: Interpretable graph neural networks with graph kernels," *arXiv preprint arXiv:2201.00491*, 2022.

[177] Y. Zhu, W. Xu, J. Zhang, Q. Liu, S. Wu, and L. Wang, "Deep graph structure learning for robust representations: A survey," *arXiv preprint arXiv:2103.03036*, 2021.

[178] L. Franceschi, M. Niepert, M. Pontil, and X. He, "Learning discrete structures for graph neural networks," in *International conference on machine learning*, 2019, pp. 1972–1982.

[179] C. Qian, G. Rattan, F. Geerts, C. Morris, and M. Niepert, "Ordered subgraph aggregation networks," *arXiv preprint arXiv:2206.11168*, 2022.

[180] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, PMLR, 2016, pp. 1050–1059.

[181] D. Slack, A. Hilgard, S. Singh, and H. Lakkaraju, "Reliable post hoc explanations: Modeling uncertainty in explainability," *Advances in Neural Information Processing Systems*, vol. 34, pp. 9391–9404, 2021.

[182] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations*, 2014.

[183] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *International conference on machine learning*, PMLR, 2014, pp. 1278–1286.

[184] K. Gregor, I. Danihelka, A. Mnih, C. Blundell, and D. Wierstra, "Deep autoregressive networks," in *International Conference on Machine Learning*, PMLR, 2014, pp. 1242–1250.

[185] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," *Advances in neural information processing systems*, vol. 28, 2015.

[186] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *International Conference on Learning Representations*, 2017.

[187] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *International Conference on Learning Representations*, 2016.

[188] G. Tucker, A. Mnih, C. J. Maddison, J. Lawson, and J. Sohl-Dickstein, "Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[189] S. Gershman and N. Goodman, "Amortized inference in probabilistic reasoning," in *Proceedings of the annual meeting of the cognitive science society*, vol. 36, 2014.

[190] C. Zhang, J. Bütepage, H. Kjellström, and S. Mandt, "Advances in variational inference," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 8, pp. 2008–2026, 2018.

[191] P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel, "The helmholtz machine," *Neural computation*, vol. 7, no. 5, pp. 889–904, 1995.