UNIVERSITY OF CALABRIA

Department of Computer Engineering, Modeling,

Electronics and Systems Engineering

PhD Program in

Information and Communication Technologies

XXXVI CYCLE

Deep Graph Representation Learning and Graph Mining for Feature-rich Networks

SSD ING-INF/05

Coordinator: Ch.mo Prof. Giancarlo Fortino

GIANCARLO FORTINO 23.05.2024 20:42:32 GMT+00:00

Supervisor: Prof. Andrea Tagarelli

ANDREA TAGARELLI 23.05.2024 22:06:40 CEST

PhD Candidate: Dott.ssa Liliana Martirano

LILIANA MARTIRANO 23.05.2024 18:13:04 GMT+01:00

To those who have valued my presence and understood my absence.
To those who are always by my side.

Preface

Rende (Italy),
February 2024

With real-world systems becoming increasingly interconnected and feature-rich, identifying network models capable of representing the rich semantics incorporated into objects and their relationships without losing information is a pivotal and demanding challenge. In recent years, complex network models and deep architectures have shown their effectiveness in learning concise representations and capturing intricate patterns, albeit targeting one or a few semantic aspects at a time.

The scope of this thesis is focused on the development of machine learning and graph mining solutions tailored to the unique challenges posed by the integration of multiple features within network structures. More specifically, this work addresses the open problems of learning node representations in a multifaceted and/or time evolving context, and unveiling the main patterns, the key actors and their relationships in complex graph structures leveraging the expressive power of feature-rich network models and deep architectures.

After a quick recall on prominent network analysis measures, graph neural network architectures and feature-rich network models as valuable reference points for subsequent analyses and a discussion on works more related to our approaches, we delve into deep graph representation learning frameworks tailored to combined feature-rich network models, and explore the integration of graph mining and text mining techniques for uncovering insights within complex social network systems.

This thesis is intended for a diverse audience comprising researchers, practitioners, and students interested in feature-rich network models, their representation and analysis. While prior knowledge of non-Euclidean data, basic graph concepts, and deep architectures is beneficial, we strive to make the content accessible to a broad audience. Whether you are a seasoned expert seeking new insights or a newcomer eager to explore this burgeoning field, we hope this work serves as a worthwhile resource in your quest for knowledge.

In concluding this preface, I wish to express my heartfelt gratitude to all those who have contributed to this work. Special thanks are due to my supervisor, *Prof. Andrea Tagarelli*, whose guidance and support have been the North Star guiding me through this academic journey, as well as to my colleagues, advisors and collaborators for their invaluable insights and discussions.

Liliana Martirano

Contents

1 Introduction 1

2 Background 7
2.1 Network analysis measures 7
2.1.1 Network-level properties 8
2.1.2 Node-level centrality measures 11
2.1.3 Transitivity measures 16
2.1.4 Mesoscopic measures 18
2.2 Graph neural networks 19
2.2.1 Graph Convolutional Networks 21
2.2.2 Graph Attention Networks 22
2.2.3 Graph Transformer Networks 24
2.2.4 GraphSAGE 25
2.2.5 Graph Isomorphism Network 26
2.3 Feature-rich networks 27
2.3.1 Heterogeneous attributed networks 28
2.3.2 Multilayer heterogeneous attributed networks 30
2.3.3 Dynamic heterogeneous attributed networks 35
2.4 Related work 36
2.4.1 Heterogeneous attributed networks 37
2.4.2 Multilayer networks 38
2.4.3 Heterogeneous attributed multilayer networks 40
2.4.4 Heterogeneous attributed networks with temporal dimension 41
2.4.5 Incremental approaches for homogeneous networks based on experience node replay 42

3 Deep graph representation learning for multilayer, heterogeneous, attributed networks 43
3.1 Introduction 43
3.2 Proposed framework 45
3.2.1 Content encoding 48

3.2.2 Graph structure encoding . . . 49
3.2.3 Final embedding based on contrastive learning . . . 59
3.3 Experimental evaluation . . . 62
3.3.1 Data . . . 62
3.3.2 Competing methods . . . 65
3.3.3 Experimental settings . . . 66
3.3.4 Results . . . 68
3.4 Concluding remarks . . . 77

4 Deep graph representation learning for dynamic, heterogeneous, attributed networks . . . 79
4.1 Introduction . . . 79
4.2 Proposed framework . . . 81
4.2.1 Identification of influenced nodes . . . 83
4.2.2 Update of the memory buffer . . . 85
4.2.3 Computational complexity aspects . . . 86
4.3 Experimental evaluation . . . 89
4.3.1 Data . . . 89
4.3.2 Advantage of HINs . . . 92
4.3.3 Competing methods . . . 93
4.3.4 Experimental settings . . . 94
4.3.5 Results . . . 94
4.4 Concluding remarks . . . 103

5 Graph mining for the evolution of social debate on climate crisis . . . 105
5.1 Introduction . . . 105
5.2 Related studies . . . 106
5.3 Data . . . 107
5.4 Proposed approach . . . 108
5.4.1 Network analysis module . . . 110
5.4.2 Topic modeling module . . . 111
5.4.3 Affective computing module . . . 112
5.5 Experimental evaluation . . . 113
5.5.1 Interaction network analysis . . . 113
5.5.2 Topic modeling . . . 120
5.5.3 Affective computing . . . 120
5.6 Concluding remarks . . . 123

6 Conclusions . . . 125

Glossary . . . 129
References . . . 131

Acronyms

BERT Bidirectional Encoder Representations from Transformers
CC Connected Component
CNN Convolutional Neural Network
Co-MLHAN Contrastive learning based framework for MultiLayer Heterogeneous Attributed Networks
COP Conferences of the Parties
c-TF-IDF class-based Term Frequency-Inverse Document Frequency
DyHANE Dynamic Heterogeneous Attributed Network Embedding
EWC Elastic Weight Consolidation
GAT Graph Attention Network
GAT-GIN Graph Isomorphism Network with Attention
GCL Graph Continual Learning
GCN Graph Convolutional Network
GCNA Graph Convolutional Network with Attention
GCN-P Graph Convolutional Network with Graph Pooling
GIN Graph Isomorphism Network
GNN Graph Neural Network
GPool-GIN Graph Isomorphism Network with Graph Pooling
GRL Graph Representation Learning
GraphSAGE Graph Sample and AGgrEgation
HIN Heterogeneous Information Network
IMDb Internet Movie Database
OVR one-vs-rest
ROC AUC Area Under the Receiver Operating Characteristic Curve
SGCN Spatial Graph Convolutional Network
UMAP Uniform Manifold Approximation and Projection
UNFCCC United Nations Framework Convention on Climate Change
VADER Valence Aware Dictionary and sEntiment Reasoner

Chapter 1
Introduction

With real-world scenarios comprising an increasing number of interconnected components, graph structures grew in popularity, and the need of complex models and concise representation of entities (nodes) and relationships (edges) connecting them arose. Network models have demonstrated their inherent ability to portray the intricate nature of real-world systems, such as social networks, collaboration/citation networks, biological networks, transportation networks, or recommendation systems, that exhibit multi-relational structures, nonlinear dependencies, temporal evolution, multi-modal characteristics.

The abstraction of tangled interconnections and properties requires adequate modeling and learning appropriate representations. Graph Representation Learning (GRL) aims to encode rich semantic and structural information of nodes, edges or (sub)graphs into a low-dimensional vector space for efficient storage and analysis. The learned representations, known as embeddings, can be used as feature input for several graph learning tasks, each offering unique insights into the underlying network structure. Downstream tasks span multiple levels of granularity, including node-, edge-, and graph-level analyses. At the node-level, tasks such as node classification, node regression and node clustering aim to predict or group nodes based on their attributes or connectivity patterns within the network. In a social network, node classification, resp. clustering, could involve predicting users' interest(s), resp. grouping users with similar interests, based on their interactions and profile information. In a collaboration network, node classification, resp. regression, could involve predicting authors' research field(s), resp. number of citations, based on their co-authorship relationships and affiliation. At the edge-level, tasks like link prediction and edge classification focus on inferring missing or future connections between nodes, crucial for understanding network evolution and predicting potential interactions. In a citation network, edge classification might involve predicting the type or strength of the relationship between academic papers based on their content or citation patterns; in a transportation network, link prediction can be used to predict future connections or beneficial roads based on traffic flow and proximity to public facilities. Finally, at the graph-level, tasks such as graph classification and community detection aim to characterize the entire network structure or identify

cohesive substructures within it. In a biological network, graph classification could involve predicting the toxicity of molecules, while community detection might identify functional modules or protein complexes within the network.

In recent year, both tailored methods optimized for a targeted task and task-independent approaches demonstrated to be effective [33], underlining the utility of graph representation learning in unveiling meaningful insights from complex network data. Complementarily, graph mining techniques play a pivotal role in uncovering hidden patterns, relationships and structures within networks of growing size and complexity. As can be guessed, most real-world network systems are plentiful in "extra" features which serve as valuable source of information and can be conveniently embedded in a network. We refer to models where the expressive power of the network topology is enhanced by exposing one or more peculiar features as *feature-rich networks* [29]. Feature-rich networks include attributed networks, heterogeneous information networks, multilayer networks, temporal or dynamic networks, and many other task-driven (like location-aware networks) and data-driven (such as probabilistic networks) models.

Attributed Networks [113] are characterized by external information content associated with nodes and/or relationships in the form of attribute vectors, encompassing numerical attributes, categorical labels, textual description and multimodal content. In several cases attribute encoding is non-trivial, including long texts associated with certain node types (e.g., plots of movies or abstracts of papers), high-definition images (e.g., movie or conference posters), videos or audios (e.g, movie trailers or conference presentations), and in presence of a mixture of textual elements, emojy, images, references and mentions typical of social network posts. The same argument applies to attributes on edges. *Heterogeneous Networks* [82] or *Heterogeneous Information Networks* (HINs) are characterized by multiple entity and/or relationship types, reflecting the diverse nature of entities and interactions and their different contributions. Furthermore, different node types may have attributes of different dimensionality. *Multilayer Networks* are characterized by replications of nodes on multiple layers according to different point of views, properties, temporal intervals or other semantics aspects, allowing for a nuanced modeling across different dimensions; they can be prominent when disentangling a composite object into its components in different layers. The presence of replicas of the same entity on multiple layers involves the processing of different contributions, since layers can account for different characteristics and priorities, and the strength of coupling between layers may vary. Furthermore, the same object in different layers can also be associated with different attributes. *Temporal* or *Dynamic Networks* [84] are characterized by changes over time in graph topology and node/edge attributes, capturing the temporal dependencies and evolution patterns. The integration of new knowledge and refining of existing knowledge enable to identify new patterns and consolidate existing patterns in the data, respectively.

Each of the aforementioned network models has been extensively studied in the literature to detect hidden patterns or key elements within the network [9, 60, 61], and to extract dense representations from sparse graph structures [29], including studies of deep graph representation learning for heterogeneous [103], heterogeneous and at-

tributed [6], or heterogeneous dynamic [26] graphs. Graph neural networks (GNNs) have demonstrated effective capability to exploit the rich information associated with these kinds of networks. Nevertheless, still under-explored is the combination of multiple network models for representation learning via GNNs and insights extraction on feature-rich networks. Traditional approaches to network analysis often simplify multi-faceted scenarios by focusing on individual aspects or subsets of the data, or are targeted toward a specific machine learning task, thereby overlooking the holistic nature of complex networks. Such simplifications can lead to an incomplete understanding of the interconnected phenomena and the underlying dynamics, and limit the effectiveness of downstream learning tasks.

Contributions

In this thesis we address research topics centered around the problem of modeling semantically rich and interconnected objects that stay tight into a unique feature-rich network model. The overall goal is to provide a unified and comprehensive view of the data, improving upon traditional models in terms of quality of results, while limiting the impact on their efficiency and scalability. Specifically, three main research topics can be identified, as detailed below.

Contrastive learning framework for multilayer heterogeneous attributed networks. The first addressed challenge is to learn node representations for static networks with rich semantics, including multiple types of nodes and relationships, additional information content associated to nodes, and replication of nodes on multiple layers, i.e, for networks that are simultaneously heterogeneous, attributed and multilayer. The proposed framework [55] learns multi-type node embeddings in an unsupervised setting, i.e., without relying on labeled data, based on a cooperative contrastive mechanism that maximizes the similarity between positive examples and minimizes the similarity between negative examples, such that similar nodes are pulled closer in the embedding space while dissimilar nodes are pushed apart. The identification of positives and negatives for each node, i.e., similar (related) and dissimilar (unrelated) data points, poses an additional challenge, due to subjectivity in similarity, data imbalance and data representation complexity. In fact, different tasks and domains may have varying degrees of similarity; in real-world datasets, similar pairs can be significantly outnumbered by negative examples; features that are important for similarity can be latent or not easily observable in the complex structure of data.

The proposed framework can be used, e.g., to learn embeddings of TV series and related directors and actors in a movie database to classify the genre(s) of specific seasons or episodes of TV series, or predict the cast of upcoming seasons or episodes. A similar approach had been defined in literature for heterogeneous but single layer networks [93], with no possibility of integrating information from the different replicas of the same entity, so without possibility of variation in granularity, e.g., discriminating episodes or seasons of the same TV series.

The research line on deep graph representation learning for multilayer, heterogeneous and attributed networks is the focus of Chapter 3.

Continual learning framework for dynamic heterogeneous attributed networks The second addressed challenge is to learn node representations for time evolving networks with rich semantics, including multiple types of nodes and relationships, additional information content associated to nodes, changes in network topology and node attributes, i.e, for networks that are simultaneously heterogeneous, attributed and dynamic. The proposed continual learning framework [52] is concerned with the encoding of changes in graph structure due to multiple events, including newly formed or deleted nodes and/or relationships, and dynamic attributes. At each new timestamp, it identifies a representative sample of multi-typed nodes as training set, including both knew knowledge — nodes affected by changes — and previous knowledge — most relevant unchanged nodes stored as experience—, and updates the parameters of a graph neural network module, in order to generate up-to-date representations for all nodes in the network. To this end, different strategies for identifying nodes affected by changes have been investigated, together with experience replay strategies for detecting the most informative unchanged nodes. In a dynamic discrete-time setting with changes in network structure and entities features, the major challenge is detecting and adapting to changes to generate effective representations for all nodes in the network. More specifically, we need to integrate new knowledge, by identifying new patterns in the data, while simultaneously refining existing knowledge, by consolidating existing patterns. Retrain the model from scratch, i.e., on the whole network, is costly, especially if only a subset of the network has been changed over time; train solely on changed nodes have the risk of losing established patterns if the new nodes exhibit unseen properties, and can led to degraded representations of unchanged nodes. This is the so-called catastrophic forgetting problem [57], i.e., the tendency of a neural network to lose proficiency in previously learned tasks when adapting to or acquiring new information.

The proposed framework can be used, e.g., to learn embeddings of papers, authors and institutions in a co-authorship - citation - affiliation network to classify the main field(s) of study of authors, or predict future collaborations. Approaches based on experience replay have been studied in the literature only for homogeneous graphs, i.e., networks with only one type of node and one type of relationship [89, 111]. Conversely, the field of study of authors is influenced by both collaborations and citations; collaborations are favored in the same research institute or related institutes, etc. Flattening to a single type of node and relationship results in loss of relevant information.

The research line on deep graph representation learning for dynamic, heterogeneous and attributed networks is the focus of Chapter 4.

Complex social networks analysis The third addressed challenge is to unveil the main patterns, the key actors and their relationships in complex social network systems, characterized by a thematically focused content — ensuring more intense interactions — and discussions featured as a discrete, regularly repeated online event for multi-year analysis. The proposed approach [54, 53] is a combination of

graph mining and text mining techniques to analyze the evolution of social debate and provide insights into the discussion by identifying the most influential users, extracting the main debated topics, assessing the evolution of users' sentiment and emotions across time, and understanding users' agreement and disagreement. The analysis concerns the monitoring of public opinion and investigation of the main trends, sentiments, and reactions underlying the online discussion on climate crisis. More specifically, the analysis was conducted on Twitter data collected during the Conferences of the Parties (COPs), the annual meetings related to the implementation of the United Nations Framework Convention on Climate Change, being the foremost global forum for multilateral discussion on climate-related matters. Social traces left by people on social platforms can provide valuable insights that might be used to make better decisions, prioritize actions, and foster more effective communication for the urgent challenge of climate change. In social networks, understanding the trend and evolution of online debate is still an open challenge, including unveiling the actors that drive the debate, deciphering information amplification patterns and addressing misinformation and disinformation challenges tracing origins and understanding propagation mechanisms. Additional challenges concern understanding the context of debates, considering ambiguous language and cultural nuances, temporal analysis to figure out topic evolution and key events shaping discussions. and assessing sentiment and emotions that distinguish the social response to the climate change narrative on social networks.

The research line on complex social network analysis is the focus of Chapter 5.

Structure of the thesis

The remainder of the thesis is structured as follows. Chapter 2 provides the necessary background on network analysis measures, advanced deep graph architectures and feature-rich network models, and discusses prominent work most related to our approach. Chapters 3, 4, and 5 delve into each research topic, discussing the details of the proposed approaches, their implementation and experimental evaluation. More specifically, Chapter 3 outlines the contrastive learning framework for multilayer heterogeneous attributed networks; Chapter 4 presents the continual learning framework for dynamic heterogeneous attributed networks; Chapter 5 proposes a combination of graph mining and text mining techniques for complex social networks analysis. Finally, Chapter 6 contains concluding remarks and provides pointers for future directions.

Chapter 2
Background

In this chapter, we delve into the foundational concepts and state-of-the-art models of deep graph representation learning and graph mining in feature-rich networks, laying the groundwork for the proposed frameworks and approaches detailed in the forthcoming chapters. More specifically, we outline some of the best-known network analysis measures for quantifying the topological properties of graphs, present the prominent advanced deep graph architectures for representation learning, introduce various categories of feature-rich networks and discuss key advancements and seminal contributions in our domain. We emphasize that our exploration is not intended to be exhaustive, but serves as a foundational primer upon which the backdrop of the research unfolds.

Given a network $G = \langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, and given a node $i \in \mathcal{V}$, we will denote with $N(i)$ the neighborhood of i, i.e., all nodes j connected to i via an edge: $N(i) = \{j \in \mathcal{V} \mid e_{ij} \in \mathcal{E}\}$ (cf. Section 2.3.1), and with $\mathcal{E}_i$ the set of edges of the subgraph induced by $N(i)$, i.e., $\mathcal{E}_i = \{e_{ij} \in \mathcal{E} \mid j \in N(i)\}$.

2.1 Network analysis measures

Network analysis is instrumental in understanding the structural properties and dynamics of complex network systems across various domains, from social networks to biological systems and infrastructure networks. The combination of multiple network analysis measures empowers to unravel the intricate patterns embedded within complex systems, thereby advancing the understanding of their behavior and facilitating informed decision-making processes.

The discussion is organized into four main parts: we first discuss network-level properties to provide fundamental insights into the overall connectivity and organization of the network; we then delve into node-level centrality measures to shed light on the roles of individuals and transitivity measures to reveal link formation dynamics; finally we discuss mesoscopic measures to investigate network cohesion.

We remind that providing a comprehensive survey is outside the scope of this work; the interested reader can refer to, e.g., [106, 94, 11], for further details.

2.1.1 Network-level properties

Network-level properties, such as density, degree distribution, average degree, average path length and diameter, offer a holistic view of the structure and characteristics of the network. Such measures provide fundamental insights into the overall connectivity and organization of the network.

In the following, we provide formal definition of density, degree distribution, average degree, degree assortativity, average path length and network diameter.

Density. Density refers to the proportion of potential connections in a network that are actually realized, i.e., how many connections exist in relation to the total possible connections within a network. For an undirected graph, with no directionality of edges, it is computed ad follows:

$$D = \frac{2 \cdot |\mathcal{E}|}{|\mathcal{V}|(|\mathcal{V}| - 1)}. \tag{2.1}$$

where $|\mathcal{E}|$ is the actual number of edges in the network and $\frac{|\mathcal{V}|(|\mathcal{V}|-1)}{2}$ is the number of possible edges in the undirected network, with $|\mathcal{V}|$ being the number of nodes.

If the graph is directed, with directionality on edges, the density computation is modified as follows:

$$D = \frac{|\mathcal{E}|}{|\mathcal{V}|(|\mathcal{V}| - 1)}. \tag{2.2}$$

since the number of possible edges in the directed network is $|\mathcal{V}|(|\mathcal{V}| - 1)$, with each edge contributing only once to the computation.

Density is a fundamental metric used to quantify the level of connectivity within a network, with high values indicating tightly connected network, where most nodes are directly or indirectly linked to each other, while low values suggest a sparse network with fewer connections between nodes. Density influences the efficiency of information flow within a network. Dense networks facilitate rapid dissemination of information or influence as there are multiple paths for transmission. In contrast, sparse networks may experience slower information diffusion due to the limited number of connections.

Degree distribution. The degree distribution refers to the probability distribution of the degrees of nodes in the network. The degree of a node is the number of connection it has to other nodes. Formally, the **degree** of a node $i \in \mathcal{V}$ is often denoted as $deg\,(i)$ and expressed as:

$$deg\,(i) = \sum_{j \in \mathcal{V}} e_{ij} \mid e_{ij} \in \mathcal{E}. \tag{2.3}$$

where e_{ij} represents an edge between node i and j.

In the case of directed edges, i.e., edges with directions, nodes have in-degree (edges pointing toward the node) and out-degree (edges pointing away from the node), denoted as $deg^{in}(i)$, $deg^{out}(i)$, respectively.

Given k the degree of a node, the degree distribution $q(k)$ represents the probability that a randomly selected node in the network has degree k. Formally:

$$q(k) = \frac{|\mathcal{V}_k|}{|\mathcal{V}|}. \tag{2.4}$$

where $|\mathcal{V}|$ is the total number of nodes in the network, and $|\mathcal{V}_k|$ is the number of nodes with degree k.

The distribution of node degrees in the network provides valuable information about the connectivity patterns and organizational properties of complex systems, helping in uncovering how nodes are interconnected and whether the network exhibits specific organizational properties such as scale-free or random connectivity.

Average degree. The average degree measures the average number of connections to a node in the network. The average degree is often denoted as $\langle deg \rangle$ and can be expressed as the average of the degree of all nodes in the network or, alternatively, as the ratio of twice the total number of edges to the total number of nodes, where the multiplication factor accounts for the contribution of each edge to the degree of two nodes. Formally:

$$\begin{aligned} \langle deg \rangle &= \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} deg(i), \\ \langle deg \rangle &= \frac{2|\mathcal{E}|}{|\mathcal{V}|}. \end{aligned} \tag{2.5}$$

where $|\mathcal{V}|$ is the total number of nodes in the network, $|\mathcal{E}|$ is the total number of edges in the network, and $deg(i)$ is the degree of node $i \in \mathcal{V}$ as defined in Eq. 2.3.

The average degree serves as a valuable metric for assessing the overall connectivity and structure of networks, reflecting the average number of connections for each node. Furthermore, it facilitates comparisons between networks of different sizes and provides insights into their behavior and evolution. A higher average degree suggests a denser network with more connections between nodes, while a lower average degree indicates a sparser network with fewer connections.

Degree assortativity. The degree assortativity quantifies the preference of nodes in a network to be connected to other nodes with similar degree. It is computed using the assortativity coefficient, which is a specific application of the Pearson coefficient for networks. The assortativity coefficient r is defined as:

$$r = \frac{\sum_{k',\hat{k}} \left(e_{k',\hat{k}} - q_{k'} q_{\hat{k}} \right)}{\sigma_q^2} \tag{2.6}$$

where $e_{k',\hat{k}}$ represents the fraction of edges that connect vertices of degrees k' and $\hat{k}$; $q_{k'}$, resp. $q_{\hat{k}}$, represents the fraction of edges that are attached to vertices of degree k', resp. $\hat{k}$; σ_q^2 is the variance of the degree distribution q.

The coefficient r ranges between -1 and 1, where positive values suggest that nodes with similar degrees tend to connect to each other (e.g., individuals tending to associate with others of similar social status or influence, while negative values indicate that nodes with dissimilar degrees tend to connect to each other (e.g., more interactions between individuals of different status or influence).

Degree assortativity provides valuable insights into the structural, functional, and dynamical properties of networks. It also helps identify core-periphery structures within networks. In assortative networks, where nodes with similar degrees connect preferentially, a core of highly connected nodes emerges, surrounded by a periphery of less connected nodes. This distinction is crucial for understanding the roles and functions of different parts of the network.

Average path length. The average path length represents the average shortest path between all pairs of nodes in the network.

The **shortest path** problem in network analysis aims to find the most efficient route between two nodes in a network. Formally, the shortest path $dist(i, j)$ between nodes i and j can be defined as:

$$dist(i, j) = \min_{p \in \mathcal{P}_{ij}} \sum_{e_{uv} \in p} c_{uv} \tag{2.7}$$

where $\mathcal{P}_{ij}$ is the set of all possible paths from node i to node j, p denotes a specific path in $\mathcal{P}_{ij}$, e_{uv} represents an edge connecting node u and v, and c_{uv} is the cost or weight associated with traversing edge e_{uv}.

The average path length is defined as the average number of steps along the shortest paths for all possible pairs of nodes in the network; formally:

$$L = \frac{1}{|\mathcal{V}|(|\mathcal{V}| - 1)} \sum_{i \neq j \in \mathcal{V}} dist(i, j). \tag{2.8}$$

where $dist(i, j)$ is the shortest path length between nodes i and j as defined in Eq. 2.7.

The average path length can measure how efficiently information or influence can spread through the network. A smaller average path length indicates that nodes in the network are closely connected, with shorter paths between them. A larger average path length suggests that nodes are more distant from each other, and more steps are needed to traverse the network.

Diameter. The diameter is the maximum shortest path length between any pair of nodes. It is defined as the maximum number of steps required to traverse the longest shortest path between any pair of nodes in the network; formally:

$$D = \max_{i,j} dist(i, j). \tag{2.9}$$

where $dist(i, j)$ is the shortest path length between nodes i and j.

The diameter provides an upper bound on how far apart the nodes in the network are from each other, measuring how connected the network is. A smaller diameter indicates that nodes in the network are more closely connected, while a larger diameter suggests a more spread-out or decentralized network.

Path-based measures, including shortest paths and network diameter, offer insights into the overall structure, connectivity and efficiency of information flow within a network, identifying critical pathways and shedding light on reachability aspects.

2.1.2 Node-level centrality measures

Centrality measures, such as degree centrality, betweenness centrality, closeness centrality, and eigenvector centrality, play a crucial role in shedding light on key actors and information flow pathways, helping to understand the importance, influence, and position of nodes within the network.

Degree centrality. Degree centrality provides insights into the importance or prominence of nodes within a network based on their connectivity. It measures the number of direct connections a node has in the network, reflecting its popularity or prominence in terms of interactions or relationships with other nodes. The degree centrality is defined for each node $i \in \mathcal{V}$ as follows:

$$C_D(i) = deg(i). \tag{2.10}$$

were $deg(i)$ is the degree of node i as defined in Eq. 2.3.

To make degree centrality independent on the graph size, the *relative* degree centrality can be defined as follows for any $i \in \mathcal{V}$:

$$\widehat{C_D}(i) = \frac{C_D(i)}{|\mathcal{V}| - 1} = \frac{deg(i)}{|\mathcal{V}| - 1}. \tag{2.11}$$

Degree centrality measures are used to identify popular nodes with greater potential to influence other nodes and quickly spread information through the network, and help identify key nodes or hubs playing a critical role in maintaining connectivity within the network. Furthermore, they provide insights into the characterization of the network structure. Networks with nodes exhibiting high degree centrality tend to have a more decentralized structure, while those with nodes exhibiting low degree centrality may be more centralized around a few key nodes.

In directed networks, *in-degree*, resp. *out-degree*, centrality can be similarly computed:

$$\begin{aligned} C_D^{in}(i) &= deg^{in}(i), \\ C_D^{out}(i) &= deg^{out}(i). \end{aligned} \tag{2.12}$$

In directed graphs, *sources* and *sinks* refer to specific types or roles of nodes within a network. Sources are nodes with only outgoing edges, i.e., nodes that do not receive any incoming edge, while sinks are nodes with only incoming edges, i.e., nodes that do not emit any outgoing edge. Formally:

$$\begin{aligned} Sources &= \{i \in \mathcal{V} \mid deg^{in}(i) = 0\}, \\ Sinks &= \{i \in \mathcal{V} \mid deg^{out}(i) = 0\}. \end{aligned} \tag{2.13}$$

where $deg^{in}(i)$, resp. $deg^{out}(i)$, denotes the in-degree, resp. out-degree, of node i.

Identifying sources and sinks is crucial for analyzing information flow, processes propagation, influence or disease spread, or resource allocation. Sources are often characterized by their ability to initiate or supply information, resources, or influence to other parts of the network, while sinks are often destinations or endpoints in the network, where information, resources, or flows are consumed or accumulated.

Closeness centrality. Closeness centrality measures how central or influential a node is in a network based on its proximity to other nodes, quantifying how close a node is to all other nodes in a network. It is calculated as the reciprocal of the sum of the shortest distances between a node and all other nodes in the network; formally:

$$C_C(i) = \frac{1}{\sum_{j \in \mathcal{V}} dist(i,j)}. \tag{2.14}$$

where $dist(i,j)$ is the shortest path between nodes i and j as computed in Eq. 2.7.

To make closeness centrality independent on the graph size, the *relative* closenesse centrality can be defined as follows for any $i \in \mathcal{V}$:

$$\widehat{C_C}(i) = (|\mathcal{V}| - 1) \cdot C_C(i) = \frac{|\mathcal{V}| - 1}{\sum_{j \in \mathcal{V}} dist(i,j)}. \tag{2.15}$$

Nodes with high closeness centrality can reach other nodes most efficiently. They tend to have shorter average distances then all other nodes in the network and can spread information or influence more rapidly throughout the network. Conversely, nodes with low closeness centrality are less efficient in reaching other nodes.

Closeness centrality helps thus identify key nodes in a network that can efficiently disseminate information or resources, and can highlight individuals who are well-connected and may serve as opinion leaders or brokers between different groups, as well as strategic hubs or intersections within the traffic flow.

Betweenness centrality. Betweenness centrality quantifies the extent to which a node lies on the shortest paths between other pairs of nodes in a network. It is calculated as the fraction of shortest paths that pass through a particular node. Formally, the betweenness centrality for a node $i \in \mathcal{V}$ is defined as:

$$C_B(i) = \sum_{j,u \in \mathcal{V}, j \neq i, u \neq i} \frac{m_{j,u}(i)}{m_{j,u}(\mathcal{V})}. \tag{2.16}$$

where $m_{j,u}(i)$ is the number of shortest paths between j and u passing through i, and $m_{j,u}(\mathcal{V})$ is the total number of shortest paths between j and u.

To make betweenness centrality independent on the graph size, the *relative* betweenness centrality can be defined as follows for any $i \in \mathcal{V}$:

$$\widehat{C_B}(i) = \frac{2 \cdot C_B(i)}{(|\mathcal{V}| - 1)(|\mathcal{V}| - 2)}. \tag{2.17}$$

where $C_B(i)$ is computed as in Eq. 2.16 and $\binom{|\mathcal{V}|-1}{2}$ is the best case, when a node is on all shortest paths between each pair of vertices.
For directed graphs, $\widehat{C_B}(i) = \frac{C_B(i)}{(|\mathcal{V}|-1)(|\mathcal{V}|-2)}$.

Nodes with high betweenness centrality act as bridges between different parts of the network. They play a critical role in facilitating communication, information flow, or resource transfer between other nodes. Removing or disrupting these nodes can significantly affect the network's connectivity and overall functioning.

Unlike closeness centrality, betweenness centrality do not require full-connectivity of the network, but its computation is resource-intensive, since requires traversing the entire network multiple times to count for each pair of nodes the number of shortest paths that pass through each intermediate node.

Eigenvector centrality. Eigenvector centrality evaluates the influence of nodes in a network by considering both their direct connections and the centrality of their neighbors, based on the principle that the importance of a node is proportional to the importance of its neighbors. It is calculated as the principal eigenvector of the adjacency matrix $\mathcal{A}_{\text{adj}}$ of the network. Formally, for a node $i \in \mathcal{V}$, it is defined as:

$$E(i) = \frac{1}{\lambda} \sum_{j \in \mathcal{V}} \mathcal{A}_{\text{adj}_{ij}} \cdot E(j) \quad . \tag{2.18}$$

where $\mathcal{A}_{\text{adj}}$ is the adjacency matrix of the network, with rows and columns correspond to nodes, and each cell representing whether there is an edge connecting the corresponding pair of vertices; λ is the dominant eigenvalue of $\mathcal{A}_{\text{adj}}$; $E(i)$, resp. $E(j)$ is the eigenvector centrality of node i, resp. j.

Representing the vector of eigenvector centralities as $\mathbf{r} \in \mathbb{R}^{|\mathcal{V}|}$, with r_i denoting the eigenvector centrality or *ranking* of i, the eigenvector centrality can be rewritten as:

$$\lambda \mathbf{r} = \mathcal{A}\mathbf{adj}^{\mathbf{T}}\mathbf{r}. \tag{2.19}$$

Eigenvector centrality provides a ranking of nodes based on their overall influence within the network. Nodes with high eigenvector centrality are connected to other highly central nodes, reflecting their ability to spread information, exert influence, or control the flow of resources throughout the network. Understanding the positions of these nodes offer insights into key players, opinion leaders, or critical points of control within the network.

Eigenvector centrality serves as the theoretical foundation for various algorithms and techniques in network analysis, which are described next. They share the evalu-

ation of the centrality of nodes based on connections to other highly central nodes, but they differ in the calculation method. These variants use iterative algorithms to update centrality scores based on network structure.

Katz centrality. Similarly to eigenvector centrality, Katz centrality measures the centrality of a node by considering the centrality of its neighbors. Conversely, the latter incorporates a parameterized sum of neighboring node centralities, allowing for more control over the influence of indirect connections. Katz centrality also introduces an *attenuation parameter* to adjust for the lower importance of longer paths between nodes. It can be defined as a weighted sum of all the powers of the adjacency matrix; formally:

$$\mathbf{r} = \sum_{k=1}^{\infty} \alpha^k (\mathcal{A}\mathbf{adj}^T)^k \mathbf{r}. \tag{2.20}$$

with $\alpha \in (0, 1)$. When α is small, Katz centrality tends to probe only the local structure of the network; as α grows, more distant nodes contribute to the centrality of a given node.

Nodes with many direct connections or connections to other highly central nodes will have higher Katz centrality scores. It captures the influence of a node's local neighborhood as well as the broader network structure. Katz centrality helps identify influential nodes that may not have many direct connections but are connected to other influential nodes.

PageRank. PageRank is a centrality measure initially developed by Google to rank web pages based on their importance in the web graph. It assigns higher scores to pages that are linked to other important pages by counting the number and quality of links they receive. PageRank is similar to eigenvector centrality in considering the centrality of neighboring nodes but differs in its specific application and iterative computation. While eigenvector centrality calculates centrality based on a single dominant eigenvector, PageRank iteratively updates centrality scores based on the centrality of neighboring nodes.

Let $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ be a matrix such that columns refer to those nodes whose status is determined by the connections received from the in-neighbor row-nodes, i.e., $\mathbf{H} = \mathbf{D}_{out}^{-1}\mathcal{A}\mathbf{adj}$, where $\mathcal{A}\mathbf{adj}$ is the adjacency matrix and $\mathbf{D}_{out}$ is a diagonal matrix storing the out-degrees of nodes. Specifically, $\mathbf{H}$ has values equal to $\frac{1}{deg^{out}(i)}$ if an edge between i and j exists, and 0 otherwise. As evident from its definition, $\mathbf{H}$ is a substochastic matrix, with rows of zeros in correspondence to sinks. Let $\mathbf{d} \in \mathbb{R}^{|\mathcal{V}|}$ be a vector indexing sinks, i.e., $\mathbf{d}(i) = 1$ if $deg^{out}(i) = 0$, and $\mathbf{d}(i) = 0$ otherwise. Given a (directed) graph G, the PageRank of nodes in G is defined as:

$$\begin{aligned} \mathbf{r} &= \alpha(\mathbf{S}^T)\mathbf{r} + (1-\alpha)\mathbf{p}, \\ \text{with} \quad \mathbf{S} &= \mathbf{H} + \frac{\mathbf{d}\mathbf{1}^T}{|\mathcal{V}|}. \end{aligned} \tag{2.21}$$

where $\alpha \in (0, 1)$ is a *dumping factor*, with higher values of α considering more link structure; $\mathbf{S}$ is the row-stochastic matrix derived from $\mathbf{H}$ with the addition of $\mathbf{d}$, such that rows corresponding to sinks sum to 1 by the insertion of *virtual* backward links representing the possibility to randomly jump to any other node in the network; $\mathbf{p}$ is the *personalization vector*, being any probability vector, by default set to $\frac{1}{|\mathcal{V}|}$, but replaceable with any vector whose non-negative components sum up to 1. It can be proportional to the score of nodes with respect to an external criterion of importance, or proportional to the contribution that nodes give to a certain topological characteristic of the network, e.g., $\mathbf{p}(i) = \frac{deg^{in}(i)}{\sum_{j \in \mathcal{V}} deg^{in}(j)}$.

PageRank iteratively updates the centrality scores based on the centrality of neighboring nodes and their outgoing links. Its computation adopts the *power iteration* method, commonly used to find the dominant eigenvector of a square matrix. Starting from any random vector $\mathbf{r}^{(0)}$ and iterating through Eq. 2.21, the algorithm converges to the PageRank scores, reflecting the importance and influence of web pages in the web graph. The power iteration method enables efficient implementation of PageRank due to the sparsity of real-world network graphs.

PageRank is used by search engines to rank web pages in search results, helping users discover relevant and authoritative content, but provides insights into the importance and influence of nodes in other applications, including social networks, citation networks, and biological networks. Nodes with many incoming links from important nodes will have higher PageRank scores.

HITS. HITS (Hyperlink Induced Topic Search)) is a centrality measure designed for analyzing hyperlink networks, such as the World Wide Web. It evaluates nodes based on their *authority* and *hub* scores, where authority represents the quality and relevance of a page's content, and hub represents its ability to link to authoritative pages. HITS differs from eigenvector centrality in evaluating nodes based on their roles as authorities or hubs rather than their overall influence within the network; like PageRank and other eigenvector centrality methods, it still apply an iterative computation of a fixed point involving eigenvector equations, but it originally sees the prestige of a page as a two-dimensional notion, thus resulting in two ranking scores for every node in the network.

The hub score of a node can be expressed as proportional to the sum of the authority scores of its out-neighbors. Analogously, the authority score of a node can be expressed as proportional to the sum of the hub scores of its in-neighbors.

Formally, given a (directed) graph G, and being $\mathbf{a} \in \mathbb{R}^{|\mathcal{V}|}$ and $\mathbf{h} \in \mathbb{R}^{|\mathcal{V}|}$ the vectors storing the authority and hub scores, respectively, the HITS equations are defined as:

$$\begin{aligned} \mathbf{a} &= \mu(\mathcal{A}\mathbf{adj}^T)\mathbf{h}, \\ \mathbf{h} &= \lambda(\mathcal{A}\mathbf{adj})\mathbf{a}. \end{aligned} \tag{2.22}$$

where μ and λ are two scaling constants needed to avoid that the authority and hub scores will grow beyond bounds.

The algorithm iteratively updates authority and hub scores based on their mutual reinforcement. Starting with initial authority and hub scores for each node, e.g., initialized uniformly or set to 1, it updates at each computation authority scores and hub scores according to Eq. 2.22 and normalize them until convergence. For each node, its authority score is updated based on the sum of hub scores of its neighboring nodes, and its hub score is updated based on the sum of authority scores of its neighboring nodes, such that nodes with many incoming links from nodes with high hub scores will have higher authority scores and nodes that link to many nodes with high authority scores will have higher hub scores, respectively. The normalization step prevents scores from becoming arbitrarily large or small during the iteration process, and ensure they represent relative importance rather than absolute values.

HITS provides insights into the structure of hyperlink networks and helps identify authoritative sources of information improving the quality of web search results, authoritative pages and influential hubs in social networks or online communities, patterns of linking behavior and relationships between nodes in various networks. Nevertheless, the assumption of identifying authorities by means of hubs might not hold in other information networks other than the Web, e.g., in citation networks, where important authors typically acknowledge other important authors. This has to some extend impacted on the less popularity of HITS with respect to PageRank, although both rates of convergence depend on the eigenvalue gap.

2.1.3 Transitivity measures

Transitivity measures provide valuable insights into the clustering or cohesion of nodes within a network, capturing the tendency of nodes to form tightly interconnected clusters or groups. They help to assess the level of connectivity within a network, shed light into link formation dynamics, unveil the extent of cohesion within social groups, provide information about the formation of cliques, subgroups, and communities.

In the following, we provide formal definition of clustering coefficient, triadic and dyadic closure.

Clustering coefficient. Clustering coefficient measures the degree to which nodes in a graph tend to cluster together, quantifying the level of local connectivity among a node's neighbors.

The clustering coefficient can be calculated for an individual node or averaged over all nodes in the network. For an individual node i the clustering coefficient $cc(i)$ is calculated as:

$$cc(i) = \frac{2 \times |\mathcal{E}_i|}{deg(i) \cdot (deg(i) - 1)}. \tag{2.23}$$

where $\mathcal{E}_i$ is the set of edges of the subgraph induced by $N(i)$, and $deg(i)$ is the degree of node i. The product of degrees represents the total number of possible edges among the neighbors of node i, except for the edge between node i and its neighbors themselves.

In simpler terms, the clustering coefficient of a node in a network measures the proportion of its neighbors that are also neighbors of each other, quantifying the likelihood that two randomly chosen neighbors of a node are connected to each other.

Local clustering coefficient. The local clustering coefficient or **Watts & Strogatz transitivity** [95] is the average of the clustering coefficients of all nodes in the network, providing a measure of how strongly connected are the neighbors of a node, i.e., the transitivity at node level. Formally, given the graph G:

$$CC^l(G) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} cc(i). \tag{2.24}$$

where $cc(i)$ is the clustering coefficient of node i computed according to Eq. 2.23.

A high clustering coefficient indicates that nodes in the network tend to form clusters or communities, while a low clustering coefficient suggests a more sparse network structure with fewer clusters.

Global clustering coefficient. Global clustering coefficient or **transitivity** is based on triplets of nodes or triads, where a triad is an ordered triple of connected vertices, connected by either two (*open triad*) or three (*closed triad*) edges. A *triangle* is a closed triad where all three nodes are directly connected to each other, forming a closed loop.

Transitivity or global clustering coefficient of a graph G is defined as the number of closed triads (or 3 x triangles) over the total number of triads, comprising both open and closed triads, as follows:

$$CC^g(G) = \frac{\#triangles \times 3}{\#triads} = \frac{\#closed\ triads}{\#triads}. \tag{2.25}$$

Transitivity measures the likelihood that two nodes connected to a common neighbor are connected themselves. In simpler terms, it measures the probability that if node i is connected to node j, and node j is connected to node u, then node i is also connected to node u. Higher values indicate that nodes tend to form tightly-knit groups or communities, while lower values suggest a more sparse network structure with fewer closed loops. While the local clustering coefficient provides information about the clustering tendency around individual nodes, the global clustering coefficient or transitivity offer a broader perspective by considering the entire network.

Reciprocity. Reciprocity measures the level of bidirectional connections or mutual relationships in a directed network. It is computed as the ratio of the number of mutual edges to the total number of edges in the network, and defined as:

$$R = \frac{|\{e_{ij} \in \mathcal{E} \mid e_{ji} \in \mathcal{E}\}|}{|\mathcal{E}|}, \qquad R = \frac{2 \times \#\text{ reciprocated edges}}{\#\text{ edges}}. \tag{2.26}$$

Reciprocity quantifies the extent to which edges in the network are reciprocated or form "two-way" connections. In simpler terms, it measures the likelihood that if node i is connected to node j, then node j is also connected back to node i. Higher values indicate a higher tendency for pairs of nodes to reciprocally form connections, that may indicate trust or influence between connected nodes, as they tend to form mutual relationships; furthermore, networks with higher reciprocity tend to be more stable and robust against changes, as mutual connections enhance information flow and cooperation.

2.1.4 Mesoscopic measures

Mesoscopic measures, such as modularity and community detection algorithms, analyze larger-scale structures within the network, unveiling the underlying organizational structures, uncovering functional modules and cohesive clusters, identifying influential communities that shape collective behaviors and core-periphery patterns. The captured properties represent an intermediate level of analysis between the microscopic (individual nodes) and macroscopic (entire network) scales.

Connected components. Strongly and weakly connected components help understand the connectivity and structure of a graph.

Strongly connected components are subsets of nodes, or subgraphs, where each node is reachable from every other node in the subset via directed edges. They represent sets of nodes that have strong connectivity among themselves, allowing for bidirectional paths between any pair of nodes within the component. Formally, a SCC is a (sub)set of nodes CC in a directed graph where for every pair of nodes i and j in CC, there is a directed path from i to j and vice versa:

$$SCC(i) = \{j \mid \mathcal{P}_{ij} \neq \emptyset \wedge \mathcal{P}_{ji} \neq \emptyset\}, \tag{2.27}$$

where $SCC(i)$ represents the strongly connected component containing node i; $\mathcal{P}_{ij}$, resp. $\mathcal{P}_{ji}$, is the set of all possible paths from node i to node j, from node j to node i, respectively.

Weakly connected components are subsets, or subgraphs, where each node is reachable from every other node in the subset via undirected edges. They represent sets of nodes that are connected in some way, either directly or indirectly, when ignoring the direction of edges. Formally, a WCC is a (sub)set of nodes CC in a directed graph where there is a path (not necessarily directed) between every pair of nodes i and J in CC:

$$WCC(i) = \{j \mid \mathcal{P}_{ij} \neq \emptyset \vee \mathcal{P}_{ji} \neq \emptyset\} \tag{2.28}$$

where $WCC(i)$ represents the weakly connected component containing node i; $\mathcal{P}_{ij}$, resp. $\mathcal{P}_{ji}$, is the set of all possible paths from node i to node j, from node j to node i, respectively.

Modularity. Modularity quantifies the degree to which the network can be partitioned into communities; i.e., it measures the extent to which the network is divided into densely connected groups with sparse connections between groups. It is commonly used to assess the quality of community detection algorithms. Formally, the modularity Q of a network partition can be computed as:

$$Q = \frac{1}{2\,|\mathcal{E}|} \sum_{ij} \left(e_{ij} - \frac{deg(i) \cdot deg(j)}{|\mathcal{E}|} \right) \delta(c_i, c_j). \tag{2.29}$$

where $|\mathcal{E}|$ is the total number of edges, e_{ij} represents the presence of an edge between nodes i and j, $deg(i)$, resp. $deg(j)$, denotes the degree of node i, resp. j; c_i, resp. c_j, is the community assignment of node i, resp. j; $\delta(c_i, c_j)$ is 1 if nodes i and j belong to the same community and 0 otherwise.

The modularity thus measures the difference between the actual number of edges within communities and the expected number of such edges in a random network with the same degree distribution. Higher values of Q indicate a denser community structure within the network partition.

2.2 Graph neural networks

Graph representation learning aims to capture the structural and relational information inherent in graphs through the creation of low-dimensional vector representations for the entire graph, subgraphs, or individual nodes and edges. More specifically, given a network $G = \langle \mathcal{V}, \mathcal{E} \rangle$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, our objective is to learn an embedding function $g : \mathcal{V} \rightarrow \mathbb{R}^d$, where d is the dimension of the latent space, with $d \ll |\mathcal{V}|$, that maps each node $i \in \mathcal{V}$ into its learned representation $\mathbf{z}_i$.

Graph representation learning approaches are conventionally categorized into traditional embedding (a.k.a. "shallow") methods and *Graph Neural Network* (GNN)-based methods. Shallow approaches conventionally involve extracting handcrafted features from the graph or its structural properties, requiring manual feature engineering and struggling to capture complex representations of graph-structured data. In contrast, GNN-based approaches offer a paradigm shift by leveraging deep learning techniques to learn representations directly from raw input data, ensuring more refined graph representations, higher flexibility in leveraging attributes at node/edge level, and generalization to unseen nodes through task-specific and node similarity based training, although at the cost of tougher memory-requirements that might

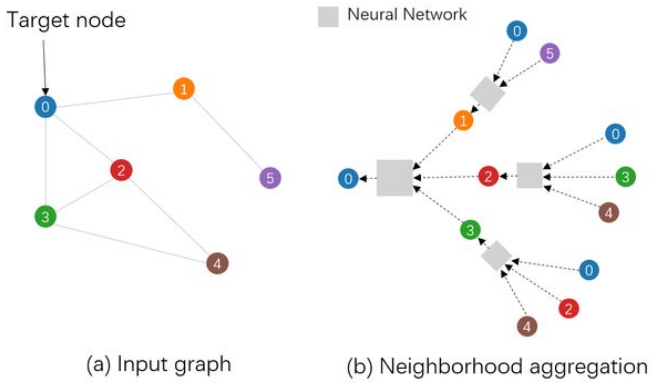


Fig. 2.1: High-level schema of message passing in graph neural networks: given an input graph (a), GNN predicts the label of the target node (e.g., the blue node) by aggregating the information from its neighboring nodes (b) [31].

impact on scalability aspects [36]. Nevertheless, they have proven to be suitable in practice for a wide range of real-world applications on large-scale graphs.

Unlike traditional neural networks designed for grid-like data, such as images or sequences, GNNs are tailored specifically for graph-structured data. Their key similarity lies in their ability to learn from data through parameterized functions and to optimize for specific objectives using gradient-based optimization techniques. However, the main difference lies in their input representation and computational framework. Traditional neural networks process fixed-size vectors or sequences, whereas GNNs operate directly on graph structures, leveraging graph topology and node features to perform message passing and aggregation across neighboring nodes, thus capturing relational information and structural dependencies inherent in graph data [75]. In the message passing paradigm, each node in the graph exchanges information with its neighbors over multiple iterations. During each iteration, a node aggregates information from its neighbors, typically by computing a weighted

Formally, given any node i in the graph, a graph neural network f (with K hidden layers) computes its embedding at each k-th layer as follows:

$$\mathbf{h}_i^{(k+1)} = f_m^{(k+1)} \left(\mathbf{h}_i^{(k)} \bigoplus \{ \mathbf{h}_j^{(k)} \mid j \in N(i) \} \right) \tag{2.30}$$

where $\mathbf{h}_i^{(0)} = \mathbf{x}_i$ is the attribute vector of node i, $\mathbf{h}_i^{(K)} = \mathbf{z}_i$ is the final learned representation for node i, and $\bigoplus$ denotes an arbitrary differentiable function aggregating feature information from the local neighborhood of nodes (e.g., summation, a pooling operator, or even a neural network [91]).

The architectures inspected below share the common foundation of message passing for information propagation across graph structures, but differ in the prop-

agation/update rule, exploring different architectural modifications, enhancements, and adaptations to address specific challenges or exploit different aspects of graph structure and feature information.

2.2.1 Graph Convolutional Networks

Graph Convolutional Networks (GCNs) extend the concept of convolutional neural networks to graph-structured data by defining convolutional operations in the spectral or spatial domain of the graph. GCNs operate by propagating information from neighbors to a central node and updating the central node's representation based on the aggregated information. The seminal work by [39] proposed a simple yet effective graph convolutional layer based on localized first-order approximations of spectral graph convolutions. In contrast to filters in Convolutional Neural Networks (CNNs), where each filter has its own unique weight matrix, the weight matrix in GCNs is unique and shared across all nodes. A notable challenge arises due to the variable number of neighbors associated with each node, unlike the fixed number of pixels in image processing. A straightforward summation of feature vectors would result in significantly larger embeddings for nodes with a higher number of neighbors. To maintain uniformity in the range of values across all nodes and ensure comparability, the result can be normalized based on node degree, i.e., on the number of formed connections. Since features originating from nodes with extensive neighbor connections tend to propagate more effectively than those from more isolated nodes, greater weights are assigned to features from nodes with fewer neighbors, thereby achieving a balanced influence across all nodes. Furthermore, was observed that stacking multiple graph layers facilitates the aggregation of information from increasingly distant nodes, but excessive layering may lead to overly intense aggregation, resulting in similar embeddings across all nodes. This phenomenon, known as over-smoothing, poses a significant challenge [42].

Formally, at each k-th layer, the GCN is defined as follows:

$$\mathbf{h}_i^{(k+1)} = \sigma\left(\sum_{j \in N^(i)} \frac{1}{\sqrt{\widetilde{\mathbf{D}}_{ii}\widetilde{\mathbf{D}}_{jj}}} \mathbf{W}^{(k)^{\mathrm{T}}} \mathbf{h}_j^{(k)}\right) \tag{2.31}$$

where $\sigma(\cdot)$ is a non-linear activation function (default is $ReLU(\cdot) = max(0, \cdot)$), $\mathbf{W}^{(k)}$ is the trainable weight matrix in the k-th convolutional layer of shape (d, d), and $\widetilde{\mathbf{D}}_{ii} = \sum_j \widetilde{\mathbf{A}}_{ij}$ is the degree matrix derived from $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$, with $\mathbf{I}_n$ as the identity matrix of size n, and n number of nodes in the graph.

The combination of local graph structure and node-level features has proved to yield good performance on node classification tasks. Neverheless, the aggregation based on the graph structure can affect the generalizability of the model. Several variants and extensions have thus been proposed.

Graph Convolutional Networks with Graph Pooling (GCN-P) enhances traditional GCNs by incorporating graph pooling operations into the GCN architecture, allowing for hierarchical feature aggregation and higher-level representations. Pooling operations aggregate information from groups of nodes or subgraphs, allowing the model to capture higher-level features and structures in the graph. GCN-P can thus learn representations at different levels of granularity, leading to improved performance on large-scale graphs and on tasks such as graph classification and clustering.

Spatial Graph Convolutional Networks (SGCNs) introduce spatial convolutions as an alternative to spectral convolutions used in traditional GCNs. Unlike traditional GCNs that operate in the spectral domain using graph Laplacian matrices, SGCNs operate directly on the spatial coordinates of nodes in the graph, enabling the model to capture geometric relationships and spatial dependencies. The incorporated graph convolutional operations that explicitly consider the spatial positions of nodes, enable more robust and interpretable representations for graphs with inherent spatial and geometric structures such as molecular graphs or 3D point clouds.

Graph Convolutional Networks with Attention (GCNAs) retain the simplicity and efficiency of traditional GCNs while introducing attention mechanisms to enhance the expressive power and flexibility of the model. Unlike other extensions based on self-attention mechanism (cf. 2.2.2), they adopt simpler attention mechanisms tailored to the specific characteristics of graph-structured data, maintaining the basic structure of GCNs and integrating attention mechanisms as additional components within each convolutional layer. GCNAs strike a balance between complexity and effectiveness, making them suitable for a wide range of graph-based tasks.

2.2.2 Graph Attention Networks

Graph Attention Networks (GATs) introduce a self-attention mechanisms into the message passing framework of GNNs. Developed by [87], GATs enable nodes to selectively attend to informative neighbors during message passing thus focusing on relevant parts of the graph. Instead of computing fixed-weighted combinations of neighbor features, GATs dynamically compute the GCN's normalization factor and the importance of each connection (attention weights) based on the compatibility between node features. Aggregation is performed by computing a weighted sum of the neighboring node features, where the weights are determined by the attention coefficients. The attention coefficients are computed to assign weights to each neighboring node based on their features' relevance to the current node. The prefix "suf" indicates that each element attends to itself as well as to other elements in the graph (similarly to words in a sentence). This means that during the attention calculation, a node considers not only the features of its neighbors but also its own features. This self-attention aspect allows the model to assign varying degrees of importance to different elements based on their contextual relevance.

The prefix "suf" signifies that, during attention computation, a node evaluates not just the features of its neighboring nodes but also its own attributes. Self-attention allows to attend to different positions within the input sequence to compute a representation of each position, i.e., each node in the neighborhood or each world in a sentence. This mechanism empowers the model to allocate differing levels of significance to individual elements, contingent upon their contextual significance, and allows for more flexible and context-aware information propagation across the network.

The embedding of a generic node i in the network with features h_i of size $\mathbb{R}^d$ is defined as:

$$\mathbf{z}_i = \sigma\Big(\sum_{j \in N(i)} \alpha_{i,j} \mathbf{W_2} \mathbf{h}_j \Big), \tag{2.32}$$

where $\sigma(\cdot)$ is the activation function (default is ELU), $\mathbf{W_2}$ is the weight matrix of shape (d, d) associated with one-hop neighbors j, $\mathbf{h}_j$ is the feature embedding of node j and $\alpha_{i,j}$ is the normalized attention coefficient indicating the importance for i of information coming from j, as defined in Eq. 2.33:

$$\begin{gathered} \alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{u \in N(i)} \exp(e_{i,u})}, \\ \text{with} \quad e_{i,j} = LeakyReLU(\mathbf{a}^{\mathrm{T}} [\mathbf{W}[\mathbf{h}_i || \mathbf{h}_j]), \end{gathered} \tag{2.33}$$

where $\mathbf{a} \in \mathbb{R}^d$ is the learnable weight vector, $[\mathbf{h}_i || \mathbf{h}_j] \in \mathbb{R}^{2d}$ is the row-wise concatenation of the column vectors associated with the two node embeddings, $\mathbf{W} = [\mathbf{W_1} || \mathbf{W_2}] \in \mathbb{R}^{d \times 2d}$ is the column-wise concatenation of $\mathbf{W_1}$ and $\mathbf{W_2}$, both of shape (d, d) and containing the left and right half of the columns of $\mathbf{W}$, associated with destination and source nodes (one-hop neighbors), respectively.

Alternatively, this operation can be carried out as $\mathbf{W}[\mathbf{h}_i || \mathbf{h}_j] = \mathbf{W_1} \mathbf{h}_i + \mathbf{W_2} \mathbf{h}_j$. Note also that in order to save the number of parameters, $\mathbf{W}$ can be constrained to $[\mathbf{W_1} || \mathbf{W_1}]$.

The self-attention mechanism can be extended similarly to [86] by employing *multi-head* attention, in order to stabilize the learning process. In this case, operations are independently replicated Q times, with different parameters, and outputs are feature-wise aggregated through an operator denoted with symbol $\bigoplus$, which usually corresponds to concatenation or average (final layer):

$$\mathbf{z_i} = \sigma\left(\bigoplus_{q=1 \ldots Q} \left(\sum_{j \in N(i)} \alpha_{i,j}^{(q)} \mathbf{W}^{(q)} \mathbf{h}_j \right) \right), \tag{2.34}$$

where $\mathbf{W}^{(q)}$ and $\alpha_{i,j}^{(q)}$ denote the weight matrix and the attention coefficient for the q-th attention head, respectively.

Each attention head learns a separate set of attention weights, enabling the model to capture diverse aspects of the relationship between nodes. By aggregating informa-

tion from multiple attention heads, complex and context-aware relationships within the graph can be captured, leading to enhanced representation learning performance.

[4] introduced additional enhancements to the graph attention mechanism, such as multi-head parameter reduction and shared parameters across different layers. The proposed GATv2 model is a dynamic graph attention variant of [87] where the order of internal operations of the scoring function is modified to apply a multi-layer perceptron for computing the score of each attended node. GATv2 aims to fix the static attention problem of standard GAT that limits its expressive power, where the ranking of attended nodes is unconditioned on the query node, in favor of improved model's expressiveness and generalization capabilities. In practice, only the order of operations of the weight calculation is altered, as follows:

$$e_{i,j} = \mathbf{a}^{\mathrm{T}}(LeakyReLU(\mathbf{W}[\mathbf{h}_i||\mathbf{h}_j])), \tag{2.35}$$

GAT and GATv2 often exhibit improved performance on tasks demanding selective attention, such as node classification or link prediction, as self-attention assigns varying levels of importance to neighboring nodes, and their efficacy encompasses a broader spectrum of graph-based applications. They eliminate the need for costly eigendecompositions demanded by spectral methods, and unlike GCNs do not require prior knowledge of the graph's structure. They are capable of operating on nodes with varying number of neighbors and on inductive learning settings, i.e., where the data learns from a training dataset and is evaluated on a testing dataset; in contrast to other optimizations of GCNs 2.2.4, GATs can process entire neighborhoods of nodes. The parallelizability of self-attention across all edges and nodes ensures rapid computation.

2.2.3 Graph Transformer Networks

Graph transformers, inspired by the success of transformers in natural language processing [86], have emerged as a powerful approach for learning representations of graph-structured data [16]. They leverage self-attention mechanisms to capture complex relationships between nodes in a graph. Unlike GATs which compute attention coefficients locally for each node based on its neighborhood, in Graph Transformers each node attends to all other nodes in the network, enabling to capture global dependencies and relationships, at the cost of computational complexity, with limitations on scalability. Graph Transformers typically incorporate positional encoding techniques to encode the relative positions of nodes in the graph, similar to how positional information is encoded in sequential data. The scaled-dot-product attention mechanism computes attention scores for a node in the network by taking the dot product of its embedding with the embeddings of other nodes, followed by scaling and softmax normalization. Formally:

$$\mathbf{z}_i = \sum_{j \in N(i)} \alpha_{i,j} \cdot \mathbf{V}_j,$$
$$\text{with} \quad \alpha_{i,j} = \text{softmax}\left(\frac{\mathbf{Q}_i \cdot \mathbf{K}_j}{\sqrt{\text{d}}}\right), \tag{2.36}$$

where $\mathbf{Q}_i$, $\mathbf{K}_j$, and $\mathbf{V}_j$ denotes the query, key and value vectors, resp., for node i and neighboring nodes j; d denotes the dimensionality of the nodes embeddings; $\alpha_{i,j}$ is the attention score, expressed using the dot product of the query and key vectors, followed by scaling to mitigate the impact of dimensionality on the magnitude of the dot product and by softmax normalization to generate the attention weights. More specifically, the query vector represents the information from node i that is used to compute attention scores with respect to its neighboring nodes, thus guiding which nodes in the neighborhood are relevant to node i; the key vector for neighboring node j encodes the information that is compared with the query vector of node i to compute the attention score, representing the importance or relevance of node j to node i; the value vector for neighboring node j holds the information content of node j, which is weighted by the attention score when aggregating information from the neighborhood. The attention mechanism focuses on relevant nodes in the neighborhood based on the similarity between query and key vectors, while the value vectors determine the information that will be aggregated and propagated to node i.

To enhance the model's ability to capture diverse relationships within the graph, similarly to multi-head attention for GATs, the scaled-dot-product attention can be applied independently in each attention head, allowing the model to attend to different parts of the input graph. The resulting attention-weighted values are then concatenated and projected back to the original dimension to obtain the final output.

2.2.4 GraphSAGE

GraphSAGE (Graph Sample and Aggregation) is a framework for scalable graph representation learning that operates in a minibatch setting, introduced by [24]. GraphSAGE addresses the challenge of training on large-scale graphs by sampling a subset of nodes and their neighborhoods for each minibatch, enabling efficient training and scalability to graphs with millions or even billions of nodes. Eeach node aggregates information from its sampled neighbors using a learned aggregation function, such as mean or max pooling. By iteratively sampling and aggregating information from different neighborhoods, GraphSAGE can capture diverse local structures within the graph and learn expressive node representations. Formally:

$$\mathbf{h}_i{}^{(k)} = \text{AGGREGATE}\left(\{\mathbf{h}_j{}^{(k-1} \mid j \in N(i)\}\right),$$
$$\mathbf{h}_i{}^{(k)} = \sigma\left(\mathbf{W}^{(k)} \cdot \text{CONCAT}\left(\mathbf{h}_i{}^{k-1}, \mathbf{h}_i\right)\right). \tag{2.37}$$

where $\mathbf{h}_i{}^{(k)}$ represents the embedding of node i at the k-th iteration, σ is the activation function, $\mathbf{W}^{(k)}$ denotes the weight matrix. The aggregation and concatenation functions can be specified on the bases of the characteristics of the network and the task at hand.

Exploring different strategies for neighborhood sampling, aggregation functions, and training objectives, multiple variants of GraphSAGE have been proposed.

Inductive GraphSAGE extends the original model to support out-of-sample node classification and inference. Unlike transductive methods that require access to the entire graph during inference, inductive methods can generalize to unseen nodes by leveraging learned node representations and neighborhood aggregation functions. Inductive GraphSAGE achieves scalability and generalization by incorporating additional parameters or techniques to learn transferable representations that capture the structural and semantic similarities between nodes in the graph.

Neighbor Sampling GraphSAGE improves the efficiency and scalability of GraphSAGE by adopting a neighbor sampling strategy during training. Instead of aggregating information from all neighbors of a node, Neighbor Sampling GraphSAGE samples a subset of neighbors for each node during each minibatch, reducing the computational overhead and memory requirements. By iteratively sampling and aggregating information from different neighborhoods, it can scale to large-scale graphs with millions or even billions of nodes while maintaining representation learning performance, thus striking a satisfying balance between computational efficiency and representation quality.

Unsupervised GraphSAGE extends the GraphSAGE framework to support unsupervised representation learning tasks such as node clustering and graph reconstruction. Unlike supervised methods that rely on labeled data for training, unsupervised methodslearn node representations solely from the graph structure and node features. Unsupervised GraphSAGE leverages self-supervised learning techniques such as graph reconstruction or node attribute prediction to learn informative and discriminative representations that capture the underlying structure of the graph.

2.2.5 Graph Isomorphism Network

Graph Isomorphism Network (GIN) is a deep learning architecture designed to learn permutation-invariant node representations, introduced by [101]. Unlike other approaches, GIN does not rely on localized neighborhood aggregation. Conversely, it aggregates information from neighboring nodes using a shared neural network function followed by a permutation-invariant readout function, ensuring that the learned node representations are consistent regardless of the ordering of nodes within the graph. Formally:

$$\mathbf{h}_i^{(k)} = \mathrm{MLP}^{(k)}\left((1+\epsilon^{(k)})\cdot\mathbf{h}_i^{(k-1)} + \sum_{j\in N(i)}\mathbf{h}_j^{(k-1)}\right) \tag{2.38}$$

where $\mathbf{h}_i(k)$ represents the embedding of node i at the k-th layer, $\text{MLP}^{(k)}$ denotes the multi-layer perceptron applied at the k-th layer, and $\epsilon^{(k)}$ is a learnable parameter vector. $\mathbf{h}_i^0 = \text{MLP}^{(0)}(\mathbf{x}_i)$, where $\mathbf{x}_i$ is the attribute vector of node i.

The key innovation lies in the permutation-invariant aggregation operation, where the embeddings of neighboring nodes are summed with the self-information, thereby allowing GIN to capture global graph structures effectively. GIN's permutation invariance and global perspective make it particularly well-suited for tasks where understanding the entire graph structure is crucial, such as graph classification and molecular property prediction. Multiple variants were proposed to improve its performance and flexibility across various graph-based tasks.

Graph Isomorphism Networks with Attention (GAT-GIN) combine the graph isomorphism network architecture with attention mechanisms inspired by GATs. By incorporating attention mechanisms into the GIN architecture, GAT-GIN enables nodes to selectively attend to informative neighbors during message passing, leading to more context-aware and expressive representation learning.

Graph Isomorphism Networks with Graph Pooling (GPool-GIN) integrate graph pooling operations into the GIN architecture to enable hierarchical representation learning. Similar to GCN-P, GPool-GIN aggregates information from groups of nodes or subgraphs using graph pooling operations, allowing the model to capture hierarchical structures and features within the graph and learn representations at different granularities, allowing for more effective modeling of complex relationships and patterns within large-scale graphs.

2.3 Feature-rich networks

The essence of feature-rich networks lies in their ability to integrate multiple information into a unified network representation. Unlike traditional networks that rely solely on structural connectivity, feature-rich networks encapsulate diverse characteristics of entities and their relationships. Any complex graph model that exposes one or more features in addition to the network topology can be defined as a feature-rich network [29], encompassing both network models that possess the capacity to incorporate supplementary information, whether at the level of nodes or edges, including attributes, multiplicity of types, or replicas reflecting distinct semantic dimensions or temporal dynamics, and models tailored to address specific hard tasks through specialized relationship modeling. Throughout the ensuing discourse, emphasis is placed on the simultaneous accommodation of multiple structural and semantic facets inherent in network data, which might enable the modeling of a spectrum of real-world scenarios. However, it is noteworthy that the discussion excludes specific interpretations of relationships, such as those associated with probabilistic network formulations.

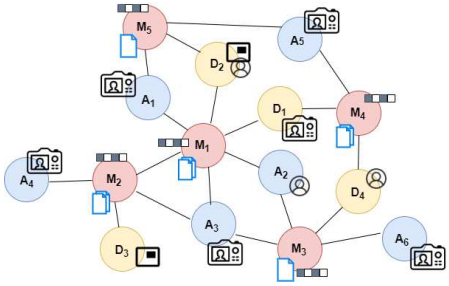


Fig. 2.2: Illustration of a heterogeneous attributed graph with three types of nodes (M (movie) - A (actor) - D (director)), and different content features for different nodes (e.g., text, images, or structured attributes).

2.3.1 Heterogeneous attributed networks

Our base feature-rich network model is a heterogeneous attributed graph, or attributed Heterogeneous Information Network (HIN), with external content associated to nodes and/or edges in the form of vector attributes, and multiple types of nodes and/or relationships.

In attributed networks, each node and/or relation can be associated with a set of attributes providing additional information on its characteristics, including numerical or categorical attributes, unstructured text, and multimedia content, such as images, videos or audios. The characteristics are encoded according to suitable methods and unified into a single vector for each node/relation. In heterogeneous networks or HINs, nodes and/or edges can be of different types, i.e., nodes and edges may represent multiple type of entities and relationships connecting them. In heterogeneous attributed networks, or attributed HINs, different node/edge types may have attributes of different dimensionality. Typically, padding or dimensionality reduction mechanisms are applied to make attribute vectors the same size, allowing all vectors to be stored in a single matrix regardless of the type; otherwise a set of matrices, one for each type, is used to keep the information separate.

Figure 2.2 depicts an example of heterogeneous attributed network on a movie dataset with three types of nodes ((M (movie) - A (actor) - D (director)), different type of relationships (e.g., A-M denotes actors starring in movies), and external content associated with each node type, such as the movies' plots, people's biographies, etc.

Formally, we define a heterogeneous attributed network or attributed HIN as $G = \langle \mathcal{V}, \mathcal{E}, A, R, \phi, \varphi, \mathcal{X} \rangle$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E} = \bigcup_{r \in R} \mathcal{E}_r \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, possibly with their weights or attributes, A is the set of node types, R is the set of relation types, $\phi : \mathcal{V} \rightarrow A$ is the node-type mapping function, $\varphi : \mathcal{E} \rightarrow R$ is the edge-type mapping function, and $\mathcal{X}$ is the single matrix or set of matrices $\mathcal{X} = \{\mathbf{X}^{(a)}\} \forall a \in A$ storing node attributes. In our formulation, we consider complex attributes on nodes stored in $\mathcal{X}$ (same size for each node type), while we consider only weights on edges, since complex attributes on edges can be treated

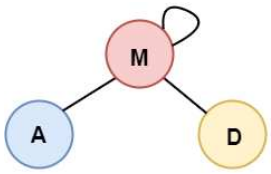


Fig. 2.3: Network schema graph corresponding to the heterogeneous attributed graph depicted in Figure 2.2.

as multiple relationships. However, the discourse can be expanded to encompass complex attributes on edges by analogously introducing a matrix, denoted as $\mathcal{X}_{\mathcal{E}}$, to store attributes on edges.

The **network schema** of a heterogeneous graph is an abstraction of the original graph showing the different node types and their direct connections. It is often referred to as meta template, since it captures node and edge type mapping functions [82]. The network schema guides the exploration of the semantics of the network. Formally, a network schema is a directed graph defined over node types A, with edges as relation types from R. Hereinafter, we refer to this graph as *network schema graph*. Figure 2.3 shows the network schema graph for the heterogeneous attributed network in Figure 2.2.

As evident from the network schema, the neighborhood of a node in a HIN is multi-typed. Given the heterogeneous graph G, we define a function, denoted as $N_r(\cdot)$, that for each node yields its neighborhood under relation type r.

To capture semantic relationships across multiple types of objects in HINs, we can look at sequences of node and edge types, known in literature as meta-paths.

A **meta-path** in a HIN is a composite relation used to model high-order proximity in the form $\sigma(a_1, a_{x+1}) = a_1 \xrightarrow{r_1} a_2 \xrightarrow{r_2} \ldots \xrightarrow{r_x} a_{x+1}$ between two node types, $a_1 \in A$ and $a_{x+1} \in A$, which are expected to share some information. Since many high-order relations can be established between node types, we denote as σ_m the $m-$meta-path type in the set $\mathcal{M}$ of all selected meta-path types. The number of nodes occurring in the meta-path yields the length of the meta-path $len(\sigma_m)$. The most informative meta-paths are usually few and of short length, and can be either identified through a hand-crafted sample by domain experts or computed through new approaches of meta-path reduction [96], interesting meta-paths mining [79] and meta-path discovery [88] even in large-scale Heterogeneous Information Networks. A *meta-path instance* is a sequence of connected nodes matching the node and edge types in the meta-path, able to make two distant nodes in the network (the *terminal* nodes) reachable. In the following, we denote as p a pair of nodes connected by at least one meta-path instance. A *meta-path-based graph* is a graph comprised of all nodes connected via meta-path instances of a determined meta-path σ_m. Given a target node type $\overline{a} \in A$, i.e., the node type targeted for a task at hand, and a meta-path σ_m with terminal nodes of the same type $\overline{a}$, the resulting meta-path-based graph is a homogeneous weighted graph with one node type $\overline{a}$ and one relation type σ_m, obtained from all meta-path instances of σ_m by removing the intermediate nodes

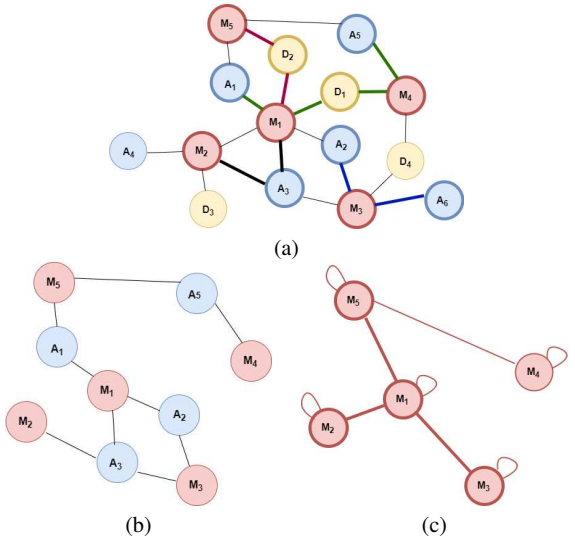


Fig. 2.4: Examples of meta-path instances for types M-A-M (movie - actor - movie) marked with bold black lines, M-D-M (movie - director - movie) marked with bold red lines, A-M-A (actor - movie - actor) marked with bold blue lines, and A-M-D-M-A (actor - movie - director - movie - actor) marked with bold green lines, w.r.t. the graph of Figure 2.2 (a), all meta-path instances of type M-A-M (b) and the corresponding meta-path based graph for M-A-M type, with focus on node M_1 and its neighbors (c).

and establishing a direct link between the pair p weighted on the number of meta-path instances connecting p. The *meta-path-based neighbors* under meta-path σ_m of a node $i \in \mathcal{V}$ is the set of nodes $N_m(i)$ connected to i via at least one meta-path instance of type σ_m. An example of meta-path types, meta-path instances and meta-path based graph is depicted in Figure 2.4.

Heterogeneous attributed graph embedding. Given a heterogeneous attributed network G, the goal is to learn an embedding function $g : \mathcal{V} \rightarrow \mathbb{R}^d$, where d is the dimension of the latent space, and $d \ll |\mathcal{V}|$. The mapping g defines the latent representation of each node $i \in \mathcal{V}$, and we use symbol $\mathbf{z}_i$ to denote its learned embedding. The learned embeddings are eventually used to support multiple downstream graph mining tasks, e.g., node classification, link prediction, etc.

2.3.2 Multilayer heterogeneous attributed networks

A *multilayer* graph is a set of interrelated graphs, each corresponding to one *layer*, with a node mapping function between any (selected) pair of layers to indicate

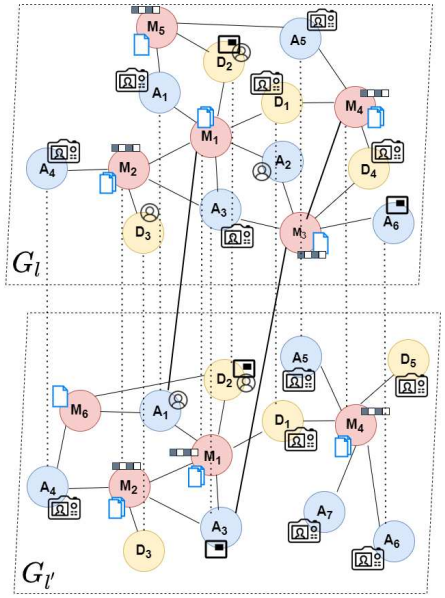


Fig. 2.5: Illustration of a multilayer heterogeneous attributed graph with two layers (G_l and $G_{l'}$), three types of nodes (M (movie) - A (actor) - D (director)), and different content features for different entities (e.g., text, images, or structured attributes). Dashed lines correspond to pillar edges, i.e, links between two instances of the same entity in different layers.

which nodes in one graph correspond to nodes in the other one. Layers can be seen as different interaction contexts, point of views, properties, temporal intervals or other semantics aspects, and we refer to the single object as *entity* and to replicas in individual layers as *entity instances* or *nodes*. Instances of the same entity are connected via *pillar-edges*. We assume that each layer can be *heterogeneous*, i.e., is characterized by nodes and/or edges of different types, such that any node can be linked to nodes of the same type as well as to nodes of different types, through the same or different relations, and is *attributed*, i.e., has nodes and/or edges associated with external information, available as set of attributes. Therefore, each layer graph has its internal set of edges, dubbed *intra-layer* or *within-layer edges*, as well as a set of edges connecting its nodes to nodes of another layer, dubbed *inter-layer* or *across-layer edges*.

Figure 2.5 depicts an example of multilayer heterogeneous attributed network on a movie dataset with two layers (G_l and $G_{l'}$), three types of entities ((M (movie) - A (actor) - D (director), with M referring to TV series comprising multiple seasons), different type of relationships (e.g., A-M denote actors starring in movies), and different external content associated with each node type, such as the movies' plots, people's biographies, etc.

Formally, we define a *multilayer heterogeneous attributed graph* as $G_{\mathcal{L}} = \langle \mathcal{L}, \mathcal{V}, V_{\mathcal{L}}, E_{\mathcal{L}}, A, R, \phi, \varphi, \mathcal{X}_{\mathcal{L}} \rangle$, where $\mathcal{L} = \{G_1, \cdots, G_\ell\}$ is the set of layer graphs, indexed in $L = \{1, \ldots, \ell\}$, with $|\mathcal{L}| = \ell \geq 2$, $\mathcal{V}$ is the set of entities, $V_{\mathcal{L}} \subseteq \mathcal{V} \times \mathcal{L}$ is the set of nodes, $E_{\mathcal{L}}$ is the set of edges, including both intra- and inter-layer edges, A is the set of entity, resp. node, types, R is the set of relation types, $\phi : \mathcal{V} \rightarrow A$ is the entity-type mapping function, $\varphi : E_{\mathcal{L}} \rightarrow R$ is the edge-type mapping function, and $\mathcal{X}_{\mathcal{L}} = \bigcup_{l=1..\ell} \mathcal{X}_l$ is the set of attribute matrices for each layer. More specifically, in each layer, it can be $\mathcal{X}_l = \{\mathbf{X}_l^{(a)}\}$, where each $\mathbf{X}_l^{(a)}$ is the attribute matrix associated with entities, resp. nodes, of type $a \in A$ in the l-th layer. Throughout this work we use symbol $\mathbf{x}_{\langle i,l \rangle}^{(a)}$ to denote the attribute vector of entity i of type a in layer G_l. We also admit that attributes can be layer-independent, in which case we indicate with $\mathbf{x}_i^{(a)}$ the attribute vector associated with entity i of type a in each layer, i.e., $\mathbf{x}_{\langle i,l \rangle}^{(a)} = \mathbf{x}_i^{(a)} \forall G_l \in \mathcal{L}$.

We specify that each entity has instances (i.e., nodes) in one or more layers, and appears at least in one layer, i.e., $\mathcal{V} = \bigcup_{l=1..\ell} \mathcal{V}_l$, with $\mathcal{V}_l$ set of entities appearing in the l-th layer. Likewise, $A = \bigcup_{l=1..\ell} A_l$, with A_l denoting the set of node types of the l-th layer, $R = \bigcup_{l=1..\ell} R_l$, with R_l denoting the set of edge types of the l-th layer, and $E_{\mathcal{L}} = \bigcup_{r \in R} E_r \subseteq V_{\mathcal{L}} \times V_{\mathcal{L}}$, with E_r indicating all the edges of type r. Moreover, $E_{\mathcal{L}}$ can be partitioned into two sets denoting the intra-layer edges and inter-layer edges. Note that inter-layer edges represent coupling structure of layers; in our setting, we assume that different coupling constraints between layers might hold, e.g., layers could be coupled with each other, only adjacent layers could be coupled, layers could follow a temporal relation order, etc. We define the set of layer pairing indices as L_{cross}, where each $\pi = (l, l') \in L_{cross}$ is a pair of coupled layers denoting an interaction between layer G_l and $G_{l'}$. We stress that in contrast to other approaches, in our formulation each layer G_l ($l = 1..\ell$) is a heterogeneous graph at both node and edge levels, i.e., $|A_l| > 1$ *and* $|R_l| > 1$. Furthermore, $A_l \subseteq A$, for all $G_l \in \mathcal{L}$, since each layer can hold a subset of the node types; $R_l \subset R$, since inter-layer connections are regarded as different types of edges.

In a multilayer heterogeneous network $G_{\mathcal{L}}$, the network schema includes all types A for individual layers and relations R, including both intra- and inter-layer edges. More specifically, we consider all relations involving any node $\langle i, l \rangle$ of target type, denoted as $R_{\langle i,l \rangle} \subseteq R$, and all node types a connected to the target node through a relation $r \in R_{\langle i,l \rangle}$.

Given the graph $G_{\mathcal{L}}$, let denote as $N_r(\cdot)$ the function that for any pair entity-layer yields its neighborhood under relation type r, regardless of the within layer or across-layer location of the neighbors. Formally, given a node $\langle i, l \rangle$, we define the set of its neighbors under relation $r \in R_{\langle i,l \rangle}$ as:

$$N_r(i, l) = \{\langle j, l' \rangle \in V_{\mathcal{L}} \,|(\langle j, l' \rangle, \langle i, l \rangle) \in E_r\}. \tag{2.39}$$

Above, note that $N_r(i, l)$ returns within-layer or across-layer neighbors of $\langle i, l \rangle$ under relation r, when $l = l'$ or $l \neq l'$, respectively.

Given a multilayer heterogeneous graph $G_{\mathcal{L}}$ and a meta-path σ_m, let $N_m(i, l)$ denote the meta-path based neighbors of node $\langle i, l \rangle$ of a certain type a, defined as

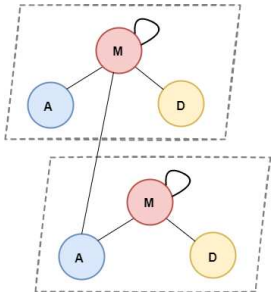


Fig. 2.6: Network schema graph corresponding to the multilayer heterogeneous attributed graph depicted in Figure 2.5.

the set of nodes of type a' that are connected with node $\langle i, l\rangle$ through at least one meta-path instance of σ_m having a as starting node-type and a' as ending node-type.

Across-layer dependencies can be modeled as particular types of meta-paths, i.e., *across-layer meta-paths*, as defined in our work [55]. They refer to the same composite relation as defined for the individual layers, with the additional constraint that the terminal nodes belong to different coupled layers, and that the intermediate node matches a pillar-edge, i.e., it corresponds to an entity with both instances involved in the composite relation. The set of across-layer meta-paths, $\mathcal{M}^{\Updownarrow}$, is defined as the the union of all meta-paths of any type and defined over all layer-pairs.

To identify the meta-path based neighbors of each node, we define two functions, denoted as $N^{\Leftrightarrow}(\cdot)$ and $N^{\Updownarrow}(\cdot)$, which for each node return the intra-layer and inter-layer neighborhood, respectively. Formally, we define the set of *within-layer* neighbors of the node $\langle i, l\rangle$, according to m-th (within-layer) meta-path type, as:

$$N_m^{\Leftrightarrow}(i, l) = \{\langle j, l\rangle \in V_{\mathcal{L}} \mid \langle j, l\rangle \in N_m(i, l)\}. \tag{2.40}$$

Similarly, the set of *across-layer* neighbors of node $\langle i, l\rangle$, according to the m-th (across-layer) meta-path type, can be defined as follows:

$$N_m^{\Updownarrow}(i, l) = \{\langle j, l'\rangle \in V_{\mathcal{L}} \mid \langle j, l'\rangle \in N_m(i, l), l' \neq l\}. \tag{2.41}$$

Note that Eqs. 2.40 and 2.41 identify the meta-path based neighborhood of type σ_m for node $\langle i, l\rangle$, with m referring to a within or across-layer meta-path, respectively; in particular, $N_m^{\Leftrightarrow}(i, l) \equiv N_m(i, l)$. Figure 2.7 depicts the extension of meta-paths to the multilayer case.

An alternative approach for modeling meta-path based graphs in a multilayer networks is still proposed by [55], via a generalization of the adjacency matrix named *supra-adjacency matrix*. The supra-adjacency matrix, denoted as $\mathbf{A}^{\text{sup}}$, has diagonal blocks each representing a layer-specific adjacency matrix (i.e., $\mathbf{A}_l \in \mathbb{R}^{n_l \times n_l}$, with $l = 1..\ell$), and off-diagonal blocks each corresponding to the inter-layer adjacency matrix $\mathbf{A}_\pi$ for layer-pair $\pi = (l, l')$, with values equal to 1 if an edge between $\langle i, l\rangle$ and $\langle j, l'\rangle$ exists, with $l \neq l'$, and 0 otherwise. In this alternative formulation, while

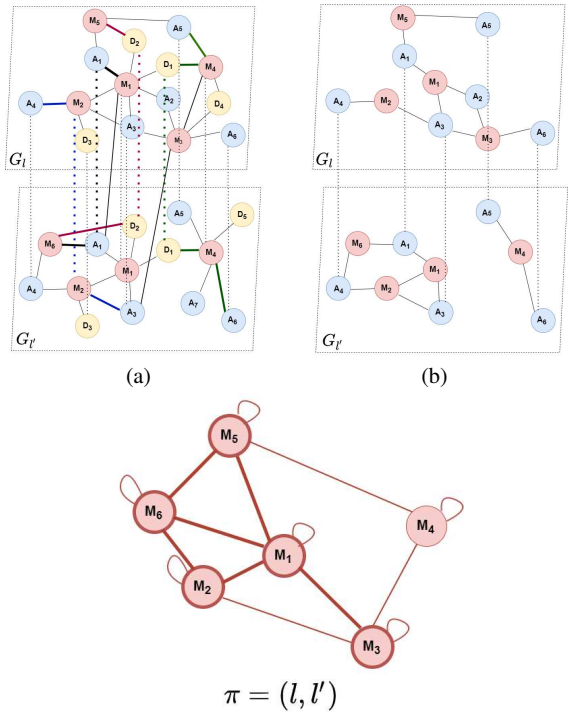


Fig. 2.7: Examples of meta-path instances for types M-A-M (movie - actor - movie) marked with bold black lines, M-D-M (movie - director - movie) marked with bold red lines, A-M-A (actor - movie - actor) marked with bold blue lines, and A-M-D-M-A (actor - movie - director - movie - actor) marked with bold green lines, w.r.t. the graph of Figure 2.5 (a), all meta-path instances of type M-A-M (b) and the corresponding meta-path based graph for M-A-M type, with focus on node M_1 and its neighbors (c).

the definition of $N^{\Leftrightarrow}(\cdot)$ does not change w.r.t. Eq. 2.40, the definition of $N^{\Updownarrow}(\cdot)$ is modified in the modeling of pillar-edges, by directly considering all the instances of the same target entities in other layers, as shown in Eq. 2.42:

$$N_m^{\Updownarrow}(i,l) = \{\langle i, l' \rangle \in V_{\mathcal{L}} \; \forall G_{l'} \in \mathcal{L} \mid l' \neq l\}. \tag{2.42}$$

Multilayer heterogeneous attributed graph embedding. Given a multilayer heterogeneous attributed network $G_{\mathcal{L}}$, the goal is to learn an embedding function at entity level $g : \mathcal{V} \rightarrow \mathbb{R}^d$, where d is the dimension of the latent space, and $d \ll |\mathcal{V}|$. Function g can be derived from an analogous function $g' : V_{\mathcal{L}} \rightarrow \mathbb{R}^d$, where d is the dimension of the latent space, and $d \ll |V_{\mathcal{L}}|$, being the embedding function at

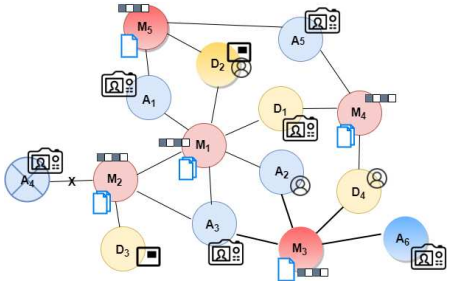


Fig. 2.8: Illustration of a dynamic heterogeneous attributed graph with three types of nodes (M (movie) - A (actor) - D (director)), and different attributes for different node types (e.g., text, images, or structured attributes) at current timestamp. Shaded nodes indicate new or changed nodes, bold edges indicate new edges, and X-marked nodes and edges indicate removal w.r.t. previous timestamp.

node level. The mapping g, resp. g', defines the latent representation of each entity $i \in \mathcal{V}$, resp. node $\langle i, l \rangle \in V_{\mathcal{L}}$, and we use symbol $\mathbf{z}_i$, resp. $\mathbf{z}_{\langle i,l \rangle}$, to denote its learned embedding. The learned embeddings are eventually used to support multiple downstream graph mining tasks, e.g., entity/node classification, link prediction, etc.

2.3.3 Dynamic heterogeneous attributed networks

Dynamic HINs or interchangeably *dynamic heterogeneous attributed graphs* are networks with multiple node and/or edge types and external information associated with nodes available as a set of attributes which evolve over time. We consider discrete network evolution comprising both changes in the graph topology, including the addition and/or removal of nodes and/or edges, and changes in attributes of isolated nodes. An example of dynamic heterogeneous network on a movie dataset is depicted in Figure 2.8, with shaded nodes indicating new or changed nodes, bold edges indicating new edges, and X-marked nodes and edges indicating removal.

Formally, we define a dynamic heterogeneous network at a generic timestamp t as $G^t = \langle \mathcal{V}^t, \mathcal{E}^t, A, R, \phi, \varphi, \mathcal{X}^t \rangle$, where $\mathcal{V}^t$ and $\mathcal{E}^t$ are the sets of nodes and edges at timestamp t, A and R are the (fixed) sets of node and relation types, with $|A|+|R| > 2$, $\phi : \mathcal{V}^t \rightarrow A$ and $\varphi : \mathcal{E}^t \rightarrow R$ are the node- and edge-type mapping functions, and $\mathcal{X}^t$ is the single matrix or set of matrices $\mathcal{X}^t = \{\mathcal{X}_a^t\} \forall a \in A$ storing node attributes at time t.

We define the discrete network evolution over time as a set of events at each timestamp t, corresponding to the set of changed edges ${\mathcal{E}_c}^t = \bigcup_{i,j} e_{ij}$, with $e_{ij} = \langle i, x_i, j, x_j, r, x_{ij}, s, c \rangle$. Each event is an edge between nodes of indices i and j of type $\varphi(e) = r$, with $\{i, j\} \in \mathcal{V}^t$; x_i and x_j denote the attribute vectors of nodes i and j resp., while x_{ij} denotes the attribute vector associated with the edge. In our

formulation, $s = 1$ (resp. $s = 0$) denotes the addition (resp. removal) of e_{ij} of type r; c denotes the *category* of the event: $c = 1$ (resp. $c = 0$) corresponds to an event with *strong* (resp. *weak*) impact on its neighbors. To handle the attribute updates of isolated nodes, i.e. nodes not involved in any change on network topology, we map the corresponding event as a self loop, with $s = 1$ and $c = 1$. We specify that each event contributes to changes in the network topology $\Delta G^t = G^t - G^{t-1}$, with ΔG^t comprising new nodes ($\mathcal{V}^t - \mathcal{V}^{t-1}$) and edges ($\mathcal{E}^t - \mathcal{E}^{t-1}$), and/or changes in the attribute matrices $\Delta \mathcal{X}^t = \mathcal{X}^t - \mathcal{X}^{t-1}$.

The network schema is assumed to be fixed over time, meaning that no new types of nodes or connections are introduced, nor are existing ones removed. However, the neighbors of a node may change over time; the dynamic neighborhood under relation r of a node at time stamp t is denoted as ${N^t}_r(\cdot)$ and defined as:

$$ {N^t}_r(i) = \{j \in \mathcal{V}^t \mid (j,i) \in \mathcal{E}_r^t\}. \tag{2.43} $$

The meta-path-based graph associated with each type is likewise dynamic. The dynamic meta-path-based neighborhood $N_m^t(i)$ of a node i is computed analogously to the dynamic neighborhood under a relation type, treating the meta-path types as different (composite) relationships. In our discrete-time setting, we define the *incremental meta-paths* for each $\sigma_m \in \mathcal{M}$ as the set of all meta-path instances of type σ_m in G^t crossing a changed edge $e \in \mathcal{E}_c^t$ [52]. A (sub)set of incremental meta-paths result in changes of the meta-path-based neighborhood of nodes — based on look-ups of the corresponding adjacency matrices $\mathcal{A}\text{adj}_m^{t-1}$ — as follows:

$$ \Delta {N^t}_m(i) = {N^t}_m(i) - {N^{t-1}}_m(i). \tag{2.44} $$

.

Dynamic heterogeneous attributed graph embedding.

Assuming a discrete set of timestamps $\{t_1, t_2, ..., t_T\}$, each of which corresponding to a set of events ${\mathcal{E}_c}^t$ determining a change in the network topology $\Delta G^t = G^t - G^{t-1}$ and/or in the attribute matrices $\Delta \mathcal{X}^t = \mathcal{X}^t - \mathcal{X}^{t-1}$, the goal is to dynamically learn an embedding function $g^t : \mathcal{V}^t \rightarrow \mathbb{R}^d$, where d is the dimension of the latent space, and $d \ll |\mathcal{V}^t|$. The mapping g is able to generate effective representations ${z_i}^t \; \forall i \in \mathcal{V}^t$, comprising both changed and unchanged nodes. The learned embeddings are eventually used to support multiple downstream graph mining tasks, e.g., node classification, link prediction, etc.

2.4 Related work

In this section we discuss most of the relevant GNN-based approaches that are designed for different feature-rich network models and particularly related to our work. Over the last years, several works focused on the extension of popular GNN models such as GCN [39] and GAT [87] to the heterogeneous or multilayer case, as well as in-

cremental GNNs for homogeneous networks. Their extension is still an open research problem. To narrow our focus, we explore in the following: semi-supervised and unsupervised learning paradigms, with emphasis on contrastive learning approaches in unsupervised contexts for multilayer and/or heterogeneous attributed networks, and GNN-based continual learning models in a discrete-time setting, with events propagating in batches at defined intervals, with emphasis on deep approaches for HINs able to incorporate the temporal dimension and incremental GNNs for homogeneous networks using a memory buffer for experience node replay.

2.4.1 Heterogeneous attributed networks

In the following we present the most prominent models for heterogeneous information networks, with explicit reference to heterogeneous approaches capable of supplementing the temporal dimension. A major challenge for heterogeneous networks is modeling information from nodes that are reachable via paths of different lengths, possibly involving different semantics and structural relations.

HetGNN [109] introduces a random walk with restart strategy to sample a fixed size of strongly correlated heterogeneous neighbors for each node, and group them on the basis of their type. It employs two modules of recurrent neural networks, encoding deep features interactions of heterogeneous contents and content embeddings of different neighboring groups, respectively, which are further combined by an attention mechanism. Co-MLHAN shares with HetGNN the modeling approach to external content encoding.

Other models leverage meta-path based neighbors and they differ in the information captured along the meta-paths. HAN [92] focuses only on the information associated with the endpoint nodes of meta-paths. It employs both node-level and semantic-level attentions. Upon the learned attention values, the model can generate node embeddings by aggregating features from meta-path based neighbors in a hierarchical manner. In addition to the information of the terminal nodes in meta-paths, MAGNN [19] also incorporates information from intermediate nodes along the meta-paths. It uses intra-meta-path aggregation to incorporate intermediate nodes, and inter-meta-path aggregation to combine messages from multiple meta-paths. DHGCN [51] incorporates both the information of the nodes along the meta-paths and the information in the ego-network of the endpoints nodes, i.e., the information coming from the direct neighbors of the terminal nodes. It utilizes a two-step schema-aware hierarchical approach, performing attention-based aggregation of information from the immediate ego-network, and attention-based aggregation of information from the neighbors of target type using meta-path based convolutions. HGT [27] takes only its direct neighbors without manually designing meta-paths but incorporating information from high-order neighbors of different types through message passing. It introduces node and edge type dependent attention mechanism and uses meta relations to parameterize the weight matrices for calculating attention over each edge. Our approach supports a user-specified selection of meta-paths and focuses

on meta-path based neighbors of specified target type. We discard the information of intermediate nodes, according to the idea of differentiating local and high-order neighborhood.

More recently developed approaches rely on considering node local and high-order structure separately. NSHE [110] introduces a network schema sampling method which generates sub-graphs (i.e., schema instances) and a multi-task learning method with different predictions to handle the heterogeneity within each schema instance, thus preserving pairwise and network schema proximity simultaneously. HeCo [93] employs a cross-view contrastive mechanism upon the definition of two views of the graph, named network schema view and meta-path view, which collaboratively supervise each other. In the network schema view, a node embedding is learned by aggregating the information from its direct neighbors, applying node-level and type-level attention for the same type and different types of nodes, respectively. In the meta-path view, a node embedding is learned by passing messages along multiple meta-paths, applying meta-path specific convolutional networks and semantic-level attention for the same and different types of meta-paths, respectively.

VACA-HINE [35] aims at jointly learning node embeddings and cluster assignments, using a variational module for the reconstruction of the adjacency matrix in a cluster-aware manner and employing multiple contrastive modules for both local and global information.

Similarly to HeCo, in our work we adopt a multi-view approach and a contrastive learning mechanism of mutual supervision between two views of the graph, with the addition of across-layer information included in the views.

2.4.2 Multilayer networks

In the following we present the most prominent models for multilayer (static) networks. Some major challenges for such kind of networks involve modeling multiple types of interactions, including both intra- and inter-layer edges, and exploiting the information of nodes matching the same entity. Here we discuss GNN-based methods focusing on their across-layer information modeling.

mGNN [22] provides a generalization of GNNs to the case of multilayer networks. It deals with outside-layer neighborhood, building an additional layer for the inter-layer relations connecting nodes in different layers. The embedding at each layer is computed propagating node features in both the intra- and inter-layer neighborhood through two independent GNN layers. We share with this approach the capability to deal with general multilayer networks with inter-layer edges not being pillar-edges. Nevertheless, unlike mGNN, Co-MLHAN can handle different types of relations in each layer.

Among the GCN-based approaches, MGCN [21] builds a graph convolutional network for each layer employing only links between nodes of the same layer, while an unsupervised term in the loss function also considers inter-layer dependencies. A different GCN-based approach is mGCN [49], which models explicit adjacency links

among different layers. mGAT [99] is an attention-based approach that introduces a regularization term to the loss function to constrain the similarity between each pair of layers. GrAMME [77] provides two different approaches, named GrAMME-SG and GrAMME-Fusion. The former explicitly builds the inter-layer edges between each node in a layer and its counterpart in a different layer, and applies a series of attention layers with the fusion-head method. The latter deals with inter-layer dependencies in a different way, as it builds layer-wise attention models and introduces an additional layer that exploits inter-layer dependencies using only fusion heads. ML-GCN and ML-GAT [107] exploit both within- and outside-layer neighborhood when computing the embedding on each layer, designing an extension of GCN and GAT architecture, resp., to multilayer networks, using the multi-head attention mechanism but without fusion-head strategy to integrate the inter-layer dependencies. Our proposed approach employs an attention-based component for learning the importance of each layer. For the modeling of intra-layer information, on the other hand, we do not exclude to use extensions of GCN or GAT, suggesting that the choice should be adapted to special needs of distinguishing between information of different importance.

More recent works introduce contrastive learning to boost the embeddings in multilayer networks. MGCCN [46] uses self-reconstruction, which learns the embedding of each layer by capturing structure and content information, and contrastive fusion, which captures the consistent information in different layers by pulling close positive pairs and pushing away negative pairs in intra-layer and inter-layer connections. Also, it exploits pillar-edges to identify positive pairs. Co-MLHAN shares the approach of allowing different attributes for nodes in different layers and of not employing data augmentation to construct negative pairs.

AMC-GNN [80] generates two graph views by data augmentation and compares the embeddings of different layers of GNN encoders to obtain feature representations, learning the importance weights of embeddings in different layers adaptively through the attention mechanism. In contrast to our approach, the two views in AMC-GNN are obtained exploiting data augmentation on the original graph. DMGI [63] integrates the relation-specific embeddings corresponding to different layers by introducing a consensus regularization framework minimizing their disagreements and a universal discriminator for all positive and negative pairs regardless of the relation type. Similar to Co-MLHAN, the views of this approach does not rely on changing the graph structure, but the similarity computation still employs a corruption of the attribute matrix, in contrast to our proposed approach. cM2NE [100] proposes a contrastive learning based embedding framework modeling multiple structural views for each layer. The contrastive learning is performed to extract information for a specific view, across the views of a layer and across the aligned layers. Our approach has a less fine-grained granularity in the multi-view mechanism, as it is not applied on each layer; on the contrary, our views include by design the across-layer information.

We would like to stress here that all the above approaches are designed for networks with only one type of node.

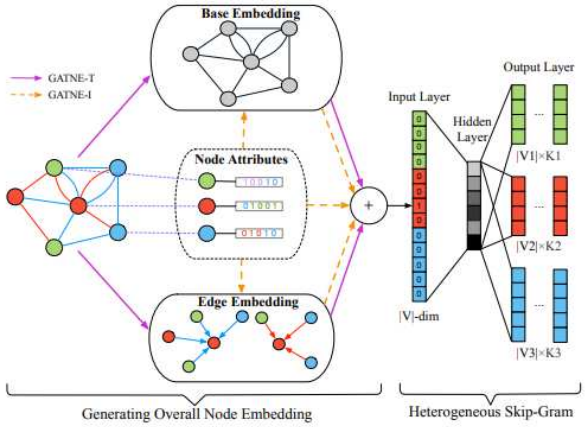


Fig. 2.9: Illustration of the workflow proposed by GATNE [5].

2.4.3 Heterogeneous attributed multilayer networks

In the past few years, interest has started to emerge in combining heterogeneity and multilayer aspects, however literature still lacks GNN-based works focusing on embedding generation for such networks. In the following, we present two models claiming to deal with networks being both heterogeneous and multilayer.

GATNE [5] splits the overall node embedding into three parts: the base embedding and attribute embedding are shared among edges of different types, while the edge embedding is computed by aggregation of neighborhood information with the self-attention mechanism. This approach uses meta-paths via meta-path based random walk strategy to generate node sequences given as input to a skip-gram model during the optimization. An overview of the framework is depicted in Figure 2.9.

MHGCN [105] includes two key learning components: multiplex relation aggregation, to distinguish the importance of the relations between different nodes, and multilayer graph convolution module, to automatically capture the short and long meta-path interactions across multi-relations by aggregating neighboring nodes' characteristics. An overview of the framework is depicted in Figure 2.10.

Our proposed approach also employs meta-paths to capture high-order relations between nodes, although the meta-path types are specified at the modeling stage, while in [5] and [105] they are automatically captured. Compared to [5], we learn a single encompassing embedding for each node/entity, incorporating different relation types; compared to [105], we learns the embeddings via a hierarchical approach that does not require decomposition and re-composition of the network. Conversely, [105] decouples the multiplex network into multiple homogeneous and bipartite sub-networks, and then re-aggregate the sub-networks with the exploration of their importance (i.e., weights) in node representation learning. Furthermore, both

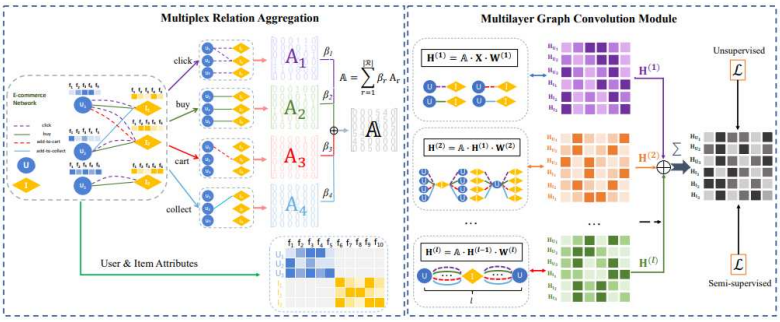


Fig. 2.10: Illustration of the workflow proposed by MHGCN [105].

methodologies described above rely on experimental evaluation employing multiplex datasets (with no possibility of modeling any inter-layer relationships other than pillar-edges) and entail one or at most two types of nodes. We want to emphasize that most existing works claiming to deal with networks being both heterogeneous and multilayer, actually refer to a multiplicity of nodes or of relations that hold globally over the network, but not necessarily on individual layers. The latter is instead an important aspect that we address in our work.

2.4.4 Heterogeneous attributed networks with temporal dimension

Existing literature on dynamic HINs commonly treats the temporal component as an additional dimension, using extra recurrent architectures or attention mechanisms for updating node representations. HDGNN [112] combines GNN and RNN architectures for scientific impact propagation learning. DYHAN [104] employs hierarchical attention for link prediction. DyHATR [102] uses a hierarchical attention model for learning static snapshots and a temporal attentive RNN for evolutionary patterns in link prediction. Co-MLHAN [55] learns node representations with hierarchical attention and a collaborative contrastive cross-view mechanism in an unsupervised setting, modeling the temporal dimension across layers. The issue inherent in these methodologies lies in their static nature, since assuming knowledge of the entire network. DyHINE [98] constructs an online update model with a dynamic operator on top of a dynamic time-series embedding model. It employs hierarchical attention to aggregate neighbor features and temporal random walks to capture dynamic interactions. In contrast to our approach, they integrate all the neighbors of changed nodes and they need to store the history of nodes, separating each embedding into past and changing embedding. More similar to our approach is LIME [66]. After mapping the nodes into a shared cuboid space and employing dynamic meta-path guided random walks and Recursive Neural Networks for initial node embeddings, it applies incremental learning to update node representations using the dynamic

Minimum Cost Maximum Flow algorithm. The update, however, is less controlled and does not directly depend on relevant changes in the neighborhood.

Differently from our approach, any of these works (except for [112]) is concerned with encoding node dynamics, such as changes in node features, while a dataset can have both static and dynamic attributes. Moreover, the approach we propose can be extended without efforts to handle removals.

2.4.5 Incremental approaches for homogeneous networks based on experience node replay

With no identified GNN-based continual learning approaches on HINs, and given the proven ability in time evolving contexts of approaches storing representative history data as experience nodes and replaying them during the training of new tasks to prevent the loss of knowledge, in the following we present the most prominent replay-based methods designed for homogeneous graphs. We share with these approaches the goal of employing incremental GNNs to support classification accuracy in the presence of changes in class distribution, based on a curated, smaller subset of network, comprising both changed and unchanged nodes. We focus on the strategy of selecting unchanged nodes to be replayed as experience.

ER-GNN [111] retains the most representative nodes of different classes in a buffer according to different selection strategies and replay them at following timestamps. The proposed strategies based on the mean of feature, coverage maximization, and influence maximization are not immediately applicable to tasks other than classification.

Conversely, similarly to our approach, ContinualGNN [89] prioritizes nodes more likely to be located at the class boundary based on their impact on the gradient, and stores those with attributes significantly distinct from their neighbors. Our approach extends this notion to diversity in attributes and local structure and to multiple node and edge types. Random-Based Rehearsal (RBR) and Priority-Based Rehearsal (PBR) [67] introduce incremental GNN models with experience replay, yielding a uniform sample of the training graph or prioritizing data points based on the model prediction error, respectively, as strategies for sample selection. In contrast to our approach, they assume a constant number of changes over time, and they do not handle changes over the node set.

Despite prioritizing the most representative nodes for updating the experience memory buffer, our proposed approach is not an active learning method. It involves the continual adaptation of machine learning models to evolving graph-structured data over time, distinguishing it from graph active learning, which specifically focuses on selecting data points for labeling.

Chapter 3
Deep graph representation learning for multilayer, heterogeneous, attributed networks

Graph representation learning has become a topic of great interest and many works focus on the generation of high-level, task-independent node embeddings for feature-rich networks with complex topology. However, the existing methods consider only few aspects of networks at a time. In this chapter, we propose a novel framework, named Co-MLHAN, to learn node embeddings for networks that are simultaneously *multilayer*, *heterogeneous* and *attributed* [55]. We leverage *contrastive learning* as a self-supervised and task-independent machine learning paradigm and define a cross-view mechanism between two views of the original graph which collaboratively supervise each other. We evaluate our framework on the entity classification task. Experimental results demonstrate the effectiveness of Co-MLHAN and its variant Co-MLHAN-SA, showing their capability of exploiting across-layer information in addition to other types of knowledge.

3.1 Introduction

Nowadays, with the ever increasing growth of interconnected data, a huge number of real-world scenarios and variety of applications can profitably be modeled using complex networks. In this context, one key aspect is how to incorporate information about the structure of the graph into machine learning models. Deep Graph Representation Learning approaches are gaining increasing attention in recent years, since they are designed to overcome the limitations of traditional, hand-engineered feature extraction methods, by learning a mapping to embed nodes, or entire (sub)graphs, as points in a low-dimensional vector space. This mapping is then optimized so that geometric relationships in this learned space reflect the structure of the original graph. After optimizing the embedding space, the learned embeddings can be used as feature inputs for downstream machine/deep learning tasks for exploration and/or prediction (e.g., node classification, community detection and evolution, link prediction).

With the emergence of GNN-based representation learning approaches, ensuring more refined graph representations, higher flexibility in leveraging attributes at node/edge level, and generalization to unseen nodes through task-specific and node similarity based training [36], efforts have been devoted to annotating graphs to provide accurate labels for learning rich representations. Since annotating graphs is costly by needing domain knowledge, *self-supervised learning* approaches are currently being investigated, which coupled with GNNs allow to learn embeddings without relying on labeled data [25].

Among different graph self-supervised learning methods, contrast-based methods have more flexible designs and broader applications compared to other approaches [47], training GNNs by discriminating positive and negative node pairs, i.e., similar and dissimilar instances. Contrastive learning aims to learn effective GNN encoders such that similar nodes are pulled together and dissimilar nodes are pushed apart in the embedding space [32].

To the best of our knowledge, there is a lack of methods able to handle networks whose nodes are replicated according to different interaction contexts or semantic aspects, are of different types and/or are connected via different types of relationships, and carry multiple information content. In other terms, networks that are simultaneously *multilayer*, *heterogeneous* and *attributed* are still unexplored in the landscape of graph representation learning, regardless of the particular learning paradigm adopted.

Contributions. To fill the above gap in the literature, in this work we propose a novel **Co**ntrastive learning based framework for **M**ulti**L**ayer **H**eterogeneous **A**ttributed **N**etworks (Co-MLHAN), which is designed to learn node/entity embeddings without relying on labeled data. Specifically, we learn node representations by contrasting positive and negative samples belonging to distinct *views* of the original graph. Inspired by recent advances in multi-view contrastive learning [25, 32, 56, 93], we indeed consider two views of a multilayer heterogeneous attributed network, which capture local and high-order (global) structure of nodes, respectively, and collaboratively supervise each other.

Our main contributions in this work correspond to addressing the following relevant, interrelated challenges:

- Representation learning for an arbitrary multilayer network such that each layer can have multiple types of nodes and relations (heterogeneous network), and have initial features associated with nodes (attributed network).
- Encoding of the *local* information of nodes, to account for the size and heterogeneity of the node neighborhoods, so as to handle variability in the number of neighbors and possible lack of certain types of neighbors.
- Encoding of the *high-order* information of nodes, by employing *meta-paths*, to reach relevant information residing multi-hops away, so that nodes of the same type that are not directly connected can be tied to each other.
- Effective integration and exploitation of *across-layer* information, including the possibility of assigning different weights to different layers or treating them equally, as needed. This also avoids using a simplistic approach based on network

flattening, so that dependencies between the layers can be retained, including both the links between the replicas of the nodes in different layers (pillar-edges) and any other inter-layer edges. Moreover, with respect to modeling the across-layer information related to pillar-edges, we also propose a variant of the main method, which will be referred to as Co-MLHAN-SA.
- Jointly learning of embeddings for each node/entity, each under the corresponding view, which can both be used for downstream tasks, such as classification. In this regard, we also provide a qualitative analysis of the interchangeability of the view-specific embeddings.
- High flexibility in terms of definition of node- and entity-level attributes as well as in terms of definition of the selection strategy of positive and negative sampling.

We experimentally evaluated our Co-MLHAN methods and selected competitors on IMDb movie data, from which we originally built multilayer heterogeneous attributed networks.

Plan of the chapter. The remainder of this chapter is structured as follows. Section 3.2 describes our proposed framework in detail. Section 3.3 provides our experimental evaluation concerning the entity classification task on IMDb network datasets, and complements a discussion on computational complexity aspects. Section 3.4 contains concluding remarks and provides pointers for future research.

3.2 Proposed framework

Co-MLHAN is a self-supervised graph representation learning approach conceived for multilayer heterogeneous attributed networks. As previously discussed, a key novelty of the proposed framework is its higher expressiveness w.r.t. existing methods, since heterogeneity is assumed to hold at both node and edge levels, possibly for each layer of the network. This capability of handling graphs that are multilayer, heterogeneous, and attributed simultaneously, enables Co-MLHAN to better model complex real-world scenarios, thus incorporating most information when generating node embeddings. In the following, we first recall the formal definition of graph representation learning in such networks, as discussed in Section 2.3.2, and provide a summary of notation used in this work in Table 3.1; then we move to a detailed description of the proposed framework.

Given a multilayer heterogeneous attributed network $G_{\mathcal{L}} = \langle \mathcal{L}, \mathcal{V}, V_{\mathcal{L}}, E_{\mathcal{L}}, A, R, \phi, \varphi, \mathcal{X}_{\mathcal{L}} \rangle$ (cf. Section 2.3.2), our goal is to learn an embedding function at entity level $g : \mathcal{V} \to \mathbb{R}^d$, where d is the dimension of the latent space, and $d \ll |\mathcal{V}|$. Function g can be derived from an analogous function $g' : V_{\mathcal{L}} \to \mathbb{R}^d$, where d is the dimension of the latent space, and $d \ll |V_{\mathcal{L}}|$, being the embedding function at node level. The mapping g, resp. g', defines the latent representation of each entity $i \in \mathcal{V}$, resp. node $\langle i, l \rangle \in V_{\mathcal{L}}$, and we use symbol $\mathbf{z}_i$, resp. $\mathbf{z}_{\langle i,l \rangle}$, to denote its learned embedding. The learned embeddings are eventually used to support multiple downstream graph

mining tasks, e.g., entity/node classification, link prediction, node regression, etc. The notation used in this work are summarized in Table 3.1.

Table 3.1: Summary of notations and their description used throughout this work.

Notations	Description
$G_{\mathcal{L}}$	Multilayer heterogeneous attributed graph
$\mathcal{L}, \ell, L, l$	Set of layers, number of layers, set of layer indexes and layer index in $G_{\mathcal{L}}$
G_l	l-th layer in $G_{\mathcal{L}}$ (single-layer heterogeneous attributed graph)
$V_{\mathcal{L}}, E_{\mathcal{L}}$	Set of nodes and set of edges in $G_{\mathcal{L}}$
$\mathcal{V}, \mathcal{V}_l, \mathcal{V}_l^{(t)}$	Set of entities in $G_{\mathcal{L}}$, set of nodes in the l-th layer, and set of entities of type t in the l-th layer
A, R	Set of node/entity types and set of relation types
A_l, R_l, n_l	Set of node types, set of relation types, and number of nodes, in the l-th layer
ϕ, φ	Node/entity and edge type mapping functions
a, t, r	Node/entity type, target entity/node type, and relation type
E_r	Set of edges of type r
d	Dimension of latent space
$\mathcal{X}_{\mathcal{L}}, \mathcal{X}_l, \mathbf{X}_l^{(a)}$	Sets of attribute matrices, layer-specific matrices, attribute matrices for entities/nodes of type a
i, j, u	Entity indexes
$\langle i,l\rangle, \langle j,l\rangle$	Node indexes (i.e., entity-layer pairs)
$L_{cross}, \pi, \delta(l,l')$	Set of layer pairing indices, pair of coupled layers, and scoring function for inter-layer links
$R_{\langle i,l\rangle}$	Set of relations involving node $\langle i,l\rangle$ of target type t
$\mathbf{x}_i^{(a)}, \mathbf{x}_{\langle i,l\rangle}^{(a)}$	Initial feature vectors for entity i and node $\langle i,l\rangle$ of type a
$\mathbf{h}_i^{(a)}, \mathbf{h}_{\langle i,l\rangle}^{(a)}$	Feature embeddings for entity i and node $\langle i,l\rangle$ of type a
$\mathbf{W}, \mathbf{b}$	Learnable weight matrix and bias term
$\mathcal{A}, \mathbf{A}_\ell, \mathbf{A}^{sup}$	Set of adjacency matrices of $G_{\mathcal{L}}$, adjacency matrix of the l-th layer, and supra-adjacency matrix
$\sigma(\cdot)$	Activation function
$\mathbf{a}$	Attention vector
α, β	Attention coefficients
$N_r(i,l)$	Set of neighbors of node $\langle i,l\rangle$ under relation r
$\mathbf{z}_{\langle i,l\rangle}^{N_r}$	Embedding of node $\langle i,l\rangle$ under relation r
Q, q	Number of attention heads and head index
$\mathbf{z}_i^{\text{NS}}, \mathbf{z}_{\langle i,l\rangle}^{\text{NS}}$	Embedding of entity i and node $\langle i,l\rangle$ under *network schema view*
$\mathcal{M}, \sigma_m, p$	Set of meta-path types, m-th meta-path type and number of (within layer) meta-path types
$\mathcal{M}^{\Updownarrow}, M_{(m,\pi)}$	Set of across-layer meta-paths and m-th across-layer meta-path type
$N_m(i,l)$	Meta-path based neighbors of node $\langle i,l\rangle$ for the m-th meta-path
$N_m^{\Leftrightarrow}(i,l), N_m^{\Updownarrow}(i,l)$	Sets of within and across neighbors of node $\langle i,l\rangle$ for the m-th meta-path
$\mathbf{z}_{\langle i,l\rangle}^{(m)}$	Embedding of node $\langle i,l\rangle$ for the m-th meta-path
$\mathbf{z}_{\langle i,\pi\rangle}^{(m)}$	Embedding of layer-pair π for the m-th across-layer meta-path
$\mathbf{z}_i^{\text{MP}}, \mathbf{z}_{\langle i,l\rangle}^{\text{MP}}$	Embedding of entity i and node $\langle i,l\rangle$ under *meta-path view*
$\hat{\mathbf{z}}_i^{\text{NS}}, \hat{\mathbf{z}}_i^{\text{MP}}$	Projected embedding of entity i under *network schema view* and under *meta-path view*
$C_{i,j}$	Number of meta-paths between entities i and j
S_i	Set of entities connected to i via a meta-path (descending order)
$T_{pos}, \overline{S}_i$	Threshold of best positives, and set of first T_{pos}-1 entities of S_i
$\mathcal{P}_i, \mathcal{N}_i$	Sets of positives and negatives for entity i
τ	Temperature parameter
$\lambda, \lambda^{\Updownarrow}, \eta$	Balancing coefficients
$L^{\text{NS}}, L^{\text{MP}}, L_{co}, L_{sup}, L_{tot}$	Loss functions

We aim to learn node embeddings in an unsupervised manner, with function g employing graph neural networks and attention mechanisms in order to encode both structural and semantic, heterogeneous and multilayer information in the context of a multi-view contrastive mechanism.

Our proposed approach is based on the *infomax* principle of maximizing mutual information [45], both in terms of graph structure encoding — complying with the distinction between local and high-order information — and across-layer information — complying with the distinction between inter-layer edges connecting direct

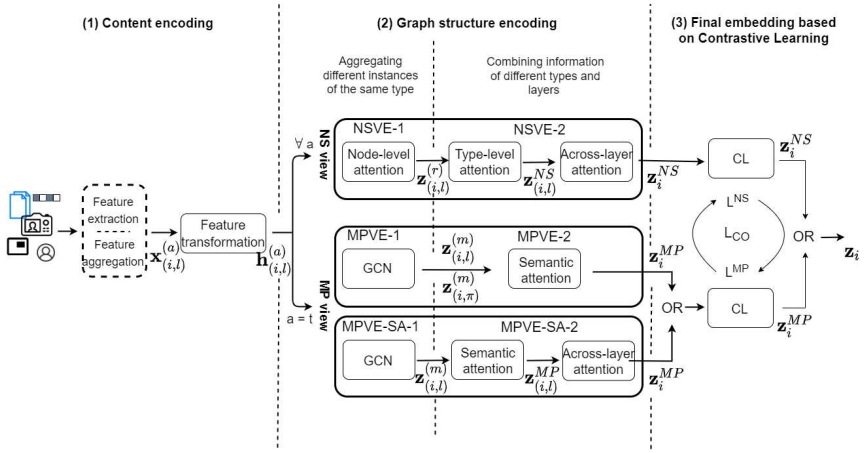


Fig. 3.1: Illustration of the three stages of the proposed Co-MLHAN framework.

neighbors and pillar-edges connecting different instances of the same entity. According to this principle, we define two different structural views on the original graph: the one is designed to encode the **local** structure of nodes and handle heterogeneity, capturing useful information from one-hop neighbors of different types (possibly from different layers), and the other one is designed to encode the **global** structure of nodes and model information from distant nodes in the network, thus capturing useful information from multi-hop neighbors of the same type. Note that we include pillar edges in the global view, since they are particular connections matching two instances of the same entity, thus enabling across-layer transitions, but they do not represent edges between two direct neighbors.

It should be emphasized that Co-MLHAN is conceived to be general and flexible, so as to exploit all available information but also being effective even when such information is lacking. For instance, across-layer relations could be limited to few replicas, nodes may show high variability in the number of neighbors, or one or more types of neighbors could be missing for some nodes.

Figure 3.1 shows a conceptual overview of our proposed framework. Accordingly, the final embedding for each target entity is learned through three main stages:

1. **Content Encoding:** Since the initial feature vectors of nodes/entities ($\mathbf{x}$) might be of different sizes, the first stage requires to transform such initial features into a shared low-dimensional latent space ($\mathbf{h}$). Moreover, this stage is also concerned with the content encoding "from scratch", i.e., generating initial embeddings from raw data associated with nodes/entities, which might be from possibly multiple and heterogeneous contents, such as categorical or numerical attributes, unstructured text and multimedia content
2. **Graph Structure Encoding:** According to the multi-view learning paradigm, the second stage requires to generate two distinct embeddings for each entity, reflecting the graph structure and maximizing the mutual information: (i) embed-

dings for the local structure ($\mathbf{z}^{\text{NS}}$), including information from all direct neighbors of the nodes being instances of the target entity, and (ii) embeddings for the high-order structure ($\mathbf{z}^{\text{MP}}$), including information from pillar-edges and from target nodes that can be reached through composite relations (i.e., meta-paths).
3. **Final Embedding based on Contrastive Learning:** The third stage requires a joint optimization between the embeddings learned under the two views to generate the final entity embedding ($\mathbf{z}$). The contrastive learning mechanism is enforced by choosing suitable positive and negative samples from the original graph.

In the following sections, we elaborate on each of the above stages. It should be noted that the graph structure encoding (stage 2) and the generation of the final embedding based on contrastive learning (stage 3) are the focus of this work. We further examine their computational complexity aspects in Section 3.3.4.4.

3.2.1 Content encoding

The first stage in our proposed framework aims to encode contents associated with nodes or entities possibly coming from external sources, which might be of different domains. Note that, for an attributed heterogeneous graph, different types of nodes could be associated with different types of content, and that even nodes of the same type could have information from multiple sources and in different forms, such as structured attributes, unstructured text, and multimedia content. External information can indeed be available either at node level or entity level, therefore initial features can be layer-dependent and associated with nodes, or layer-independent and associated with entities.

As introduced in Section 2.3.2, given a type $a \in A$, we denote with $\mathbf{x}^{(a)}_{\langle i,l\rangle}$ the initial feature vector of node $\langle i, l\rangle$ (i.e., entity i in layer G_l), and with $\mathbf{x}^{(a)}_i$ the initial feature vector of entity i. We admit that the initial feature vectors corresponding to different entity/node types could be of different lengths. If this should hold, the content encoding stage would require a feature transformation step in order to project features of different types to the same latent space, using type-specific transformation matrices. Formally, in case of content-features associated with entities, we obtain the projected feature embedding $\mathbf{h}^{(a)}_i$, for entity i of type a, as follows:

$$\mathbf{h}^{(a)}_i = \sigma(\mathbf{W}^{(a)}\mathbf{x}^{(a)}_i + \mathbf{b}^{(a)}), \tag{3.1}$$

where $\mathbf{W}^{(a)} \in \mathbb{R}^{d \times d^{(a)}_{in}}$ and $\mathbf{b}^{(a)} \in \mathbb{R}^d$ are the learnable matrix and bias term for the entity type a, respectively, and $\mathbf{x}^{(a)}_i$ is the initial feature vector of length $d^{(a)}_{in}$ associated with entity i. Analogously, in case of content-features associated with nodes, i.e., dependent on the specific layer, we obtain the projected feature embedding $\mathbf{h}^{(a)}_{\langle i,l\rangle}$, for node $\langle i, l\rangle$ of type a, as follows:

$$\mathbf{h}_{\langle i,l\rangle}^{(a)} = \sigma(\mathbf{W}_l^{(a)}\mathbf{x}_{\langle i,l\rangle}^{(a)} + \mathbf{b}_l^{(a)}), \tag{3.2}$$

where $\mathbf{W}_l^{(a)}$ and $\mathbf{b}_l^{(a)}$ are the learnable layer-specific matrix and bias term for the entity type a, respectively, and $\mathbf{x}_{\langle i,l\rangle}^{(a)}$ is the initial feature vector of length $d_{in}^{(a)}$ associated with node $\langle i,l\rangle$.

For both Eqs. 3.1 and 3.2, $\sigma(\cdot)$ is a non-linear activation function; by default, we define it as $ELU(\cdot) = max(0,\cdot) + min(0, \mu\exp(\cdot) - 1)$, with $\mu = 1$. Note also that d is chosen such that $d \leq min_{a\in A}\{d_{in}^{(a)}\}$.

Considering the possibility that each entity/node, regardless of its type, could be associated with information coming from multiple and diverse sources, the process of content feature generation would be more articulated as two aspects should be considered, namely *content-specific feature extraction* and *multi-modal content feature aggregation*. Indeed, an aggregation step would be needed to integrate contents from different modalities (i.e., structured attributes, text, images, etc.), and it can effectively be carried out by supplying an autoencoder model with the concatenation of the various content-specific embeddings, or by using an attention layer for their convex combination. Moreover, the aggregation step would be preceded by content-specific feature extraction in case the feature vectors $\mathbf{x}$ were not immediately available, and hence suitable methods (e.g., word embeddings or contextualized language models for text, convolutional networks for images, etc.) should be applied to generate features from the raw data associated with nodes/entities.

We also allow that each entity/node, regardless of its type, could be associated with no external information; in this case, initial features could be randomly generated, using identity matrices or sampling from a selected type of distribution (e.g., uniform, normal, exponential). It should however be noted that content feature generation is beyond the objectives of this work; the interested reader can refer to recently developed literature on this topic, such as [2] which proposes a general self-supervised learning framework for generating contextualized latent representation of different modalities, including speech, images and text.

3.2.2 Graph structure encoding

The second stage models two graph views, named *network schema view* and *meta-path view*, able to encode the local and global structure surrounding nodes, respectively, while exploiting multilayer information. As defined in Section 2.3.1, the network schema of a multilayer heterogeneous network is an abstraction of the original graph showing the different node types and their direct connections, comprising both intra- and inter-layer edges; meta-paths in multilayer heterogeneous networks are composite relationships connecting distant nodes in the network via sequences of nodes, with terminal and intemediate(s) nodes in the same layer (within-layer meta-paths) or terminal nodes in different layers and the intermediate node matching two different instances of the same node. Following [93], given a target entity, the

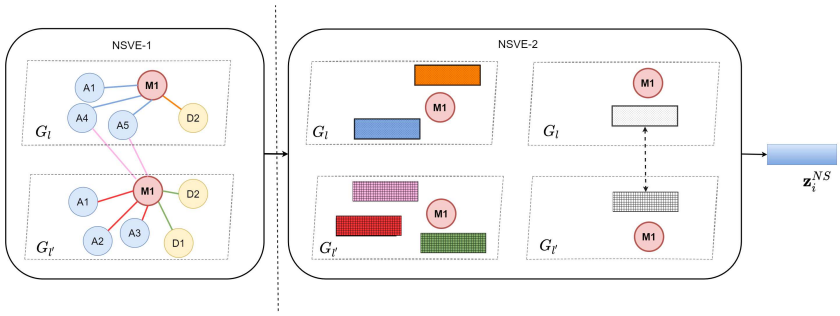


Fig. 3.2: Illustration of hierarchical attention approach used in the steps of the network schema view embedding (i.e., NSVE-1 and NSVE-2), with focus on the target entity M_1 in different layers. From left to right, NSVE-1 box shows node-level attention w.r.t. target nodes M_1, with colored lines denoting different relations and matching different attention weights; NSVE-2 box shows type-level attention w.r.t. target nodes M_1, with different colors, resp. textures, denoting the embeddings obtained from different relations, resp. layers, and across-layer attention w.r.t. target nodes M_1, combining the embeddings of different layers.

network schema view is used to capture the local structure, by modeling information from all the direct neighbors of the corresponding target nodes, whereas the meta-path view is used to capture the global structure, by modeling information from all the nodes connected to the corresponding target nodes through a meta-path and from the pillar-edges derived by the corresponding meta-path based graph.

View embedding generation. The two views exploit features associated with different entity types; specifically, the network schema view takes advantage of features of neighbors of any type, while the meta-path view takes advantage of features of nodes of target type involved in high order relations.

We remind that Co-MLHAN produces for each target entity a distinct embedding under each view. Nonetheless, both views share two fundamental steps in the embedding generation: (i) aggregating information of different instances of the same type — i.e., instances of the same relation and instances of the same meta-path, respectively — and (ii) combining information of different types — i.e., different types of relations and of meta-paths, respectively, as well as different layers.

3.2.2.1 Network schema view embedding

In the network schema view, the embedding of each target node is computed from its direct neighbors, both within and across layers. As mentioned before, the network

schema is a multilayer heterogeneous graph, having nodes of different types and relations corresponding to intra- and inter-layer edges involving nodes of target type.

To generate the embeddings under the network schema view, we follow a *hierarchical* attention approach, consisting of two main steps, which are summarized as follows and depicted in Figure 3.2:

(NSVE-1) First, we aggregate information of the same type (i.e., different instances of the same relation type) via *node-level* attention, learning the importance of each neighbor and obtaining, for each node, an embedding w.r.t. each relation type that involves a node of target type t.

(NSVE-2) Second, we combine information of different types (i.e., different relations in different layers) via *type-level* attention, learning the most relevant relations and obtaining an embedding for each node under the network schema view. Moreover, we combine information from different layers via *across-layer* attention, learning the importance of each layer and obtaining, for each entity, a single embedding under the network schema view.

Note that we refer to relation type and not to node type to be consistent in the event that target nodes are connected to a certain node type through multiple relationships. We point out that, in accordance with the infomax principle, the network schema view does not model pillar-edges, since they are processed in the other view. We also specify that intra-layer edges in different layers are seen as different types of relations, reflecting the separation into layers according to a certain aspect. In practice, layers are an additional way for distinguishing the context of relations.

Aggregating information of different instances of the same type (NSVE-1).

Aggregating information of the same type (i.e., different instances of the same relation type) takes place via *node-level attention*. This step exploits features of nodes connected to target nodes through a direct link, whether they are of the same type as the target or not.

As defined in Section 2.3.2, given the graph $G_{\mathcal{L}}$, the function $N_r(\cdot)$ yields for any pair entity-layer its neighborhood under relation type r, regardless of the within layer or across-layer location of the neighbors. More specifically, $N_r(i,l)$ returns within-layer or across-layer neighbors of $\langle i,l\rangle$ under relation r, when $l = l'$ or $l \neq l'$, respectively. (Recall that pillar edges are excluded from the definition of neighbor sets). Moreover, to ensure the aggregation of the same amount of information, we sample a fixed size of neighbors to be processed at each epoch by setting a threshold value for each type of neighbor (cf. Section 3.3.3). In our setting, neighbor sampling can be done with and without replacement. Note that this neighbor sampling approach allows for saving computational resources in case of huge networks.

We thus define the embedding of entity i in layer G_l based on neighbors under relation r as:

$$\mathbf{z}_{\langle i,l\rangle}^{N_r} = \sigma\Big(\sum_{\langle j,l'\rangle \in N_r(i,l)} \alpha_{\langle i,l\rangle,\langle j,l'\rangle}^{(r)} \mathbf{W_2}^{(r)} \mathbf{h}_{\langle j,l'\rangle}\Big), \tag{3.3}$$

where $\mathbf{z}_{\langle i,l\rangle}^{N_r}$ is the embedding of node $\langle i,l\rangle$ obtained from neighborhood under relation r, $\sigma(\cdot)$ is the activation function (default is ELU), $\mathbf{W_2}^{(r)}$ is the weight matrix of shape (d,d) associated with one-hop neighbors $\langle j,l'\rangle$, $\mathbf{h}_{\langle j,l'\rangle}$ is the feature embedding of node $\langle j,l'\rangle$ and $\alpha_{\langle i,l\rangle,\langle j,l'\rangle}^{(r)}$ is the normalized attention coefficient for the relation r connecting $\langle i,l\rangle$ and $\langle j,l'\rangle$ and indicating the importance for $\langle i,l\rangle$ of information coming from $\langle j,l'\rangle$, as defined in Eq. 3.4:

$$\begin{aligned} \alpha_{\langle i,l\rangle,\langle j,l'\rangle}^{(r)} &= \frac{\exp(e_{\langle i,l\rangle,\langle j,l'\rangle}^{(r)})}{\sum_{\langle u,l'\rangle \in N_r(i,l)} \exp(e_{\langle i,l\rangle,\langle u,l'\rangle}^{(r)})}, \\ \text{with} \quad e_{\langle i,l\rangle,\langle j,l'\rangle}^{(r)} &= \mathbf{a}^{(r)^{\mathrm{T}}}(LeakyReLU(\mathbf{W}^{(r)}[\mathbf{h}_{\langle i,l\rangle}||\mathbf{h}_{\langle j,l'\rangle}])), \end{aligned} \tag{3.4}$$

where $\mathbf{a}^{(r)} \in \mathbb{R}^d$ is the learnable weight vector under relation r, $[\mathbf{h}_{\langle i,l\rangle}||\mathbf{h}_{\langle j,l'\rangle}] \in \mathbb{R}^{2d}$ is the row-wise concatenation of the column vectors associated with the two node embeddings, $\mathbf{W}^{(r)} = [\mathbf{W_1}^{(r)}||\mathbf{W_2}^{(r)}] \in \mathbb{R}^{d\times 2d}$ is the column-wise concatenation of $\mathbf{W_1}^{(r)}$ and $\mathbf{W_2}^{(r)}$, both of shape (d,d) and containing the left and right half of the columns of $\mathbf{W}^{(r)}$, associated with destination and source nodes (one-hop neighbors), respectively.[1] In Eq. 3.4, we adopt the same approach as in GATv2 [4], which aims to fix the static attention problem of standard Graph Attention Network (GAT) [87] that limits its expressive power, since the ranking of attended nodes is unconditioned on the query node; on the contrary, GATv2 is a dynamic graph attention variant where the order of internal operations of the scoring function is modified to apply an MLP for computing the score of each attended node.

The self-attention mechanism can be extended similarly to [86] by employing *multi-head* attention, in order to stabilize the learning process. In this case, operations are independently replicated Q times, with different parameters, and outputs are feature-wise aggregated through an operator denoted with symbol $\bigoplus$, which usually corresponds to average (default) or concatenation:

$$\mathbf{z}_{\langle i,l\rangle}^{N_r} = \sigma\left(\bigoplus_{q=1\ldots Q}\left(\sum_{\langle j,l'\rangle \in N_r(i,l)} \alpha_{\langle i,l\rangle,\langle j,l'\rangle}^{(r,q)} \mathbf{W}^{(r,q)} \mathbf{h}_{\langle j,l'\rangle}\right)\right), \tag{3.5}$$

where $\mathbf{W}^{(r,q)}$ and $\alpha_{\langle i,l\rangle,\langle j,l'\rangle}^{(r,q)}$ denote the weight matrix and the attention coefficient for the q-th attention head under relation r, respectively.

[1] Alternatively, this operation can be carried out as $\mathbf{W}^{(r)}[\mathbf{h}_{\langle i,l\rangle}||\mathbf{h}_{\langle j,l'\rangle}] = \mathbf{W_1}^{(r)}\mathbf{h}_{\langle i,l\rangle} + \mathbf{W_2}^{(r)}\mathbf{h}_{\langle j,l\rangle}$. Note that in order to save the number of parameters, $\mathbf{W}^{(r)}$ can be constrained to $[\mathbf{W_1}^{(r)}||\mathbf{W_1}^{(r)}]$.

Let $\mathbf{z}_{\langle i,l\rangle}^{N_r}$ be the embedding of a target node $\langle i,l\rangle$ obtained from its neighbors in each layer under relation r. Downstream of node-level attention, we thus obtain $\bigcup_{r\in R_{\langle i,l\rangle}} \{\mathbf{z}_{\langle i,l\rangle}^{N_r}\}$ embeddings.

Combining information of different types and layers (NSVE-2).

In order to combine information of different node types according to the different relations with target nodes, we employ *type-level attention* for each layer separately. For each target node $\langle i,l\rangle$, we obtain the embedding under the network schema view $\mathbf{z}_{\langle i,l\rangle}^{\mathrm{NS}}$, as defined in Eq. 3.6:

$$\mathbf{z}_{\langle i,l\rangle}^{\mathrm{NS}} = \sum_{r\in R_{\langle i,l\rangle}} \beta^{(r)} \mathbf{z}_{\langle i,l\rangle}^{N_r}, \tag{3.6}$$

where

$$\begin{gathered} \beta^{(r)} = \frac{\exp\left(w^{(r)}\right)}{\sum_{r'\in R_{\langle i,l\rangle}} \exp\left(w^{(r')}\right)}, \\ \text{with } \; w^{(r)} = \frac{1}{|\mathcal{V}_l^{(t)}|} \sum_{\langle i,l\rangle\in\mathcal{V}_l^{(t)}} \mathbf{a}^{\mathrm{NS}^{\mathrm{T}}} \tanh(\mathbf{W}^{\mathrm{NS}} \mathbf{z}_{\langle i,l\rangle}^{N^{(r)}} + \mathbf{b}^{\mathrm{NS}}), \end{gathered} \tag{3.7}$$

where $\mathcal{V}_l^{(t)}$ is the set of entities of target type t in layer l; $\mathbf{a}^{\mathrm{NS}} \in \mathbb{R}^d$ is the type-level attention vector; $\mathbf{W}^{\mathrm{NS}}$ and $\mathbf{b}^{\mathrm{NS}}$ are the learnable weight matrix and the bias term, respectively, under the network schema view, shared by all relation types. We hence obtain the set of embeddings $\bigcup_{l\in L} \{\mathbf{z}_{\langle i,l\rangle}^{\mathrm{NS}}\}$ under the network schema view for each target node.

In order to map the learned node embeddings into the same space of the contrastive loss function, we apply an additional level of attention, i.e., *across-layer attention*. This is designed to evaluate the importance of each layer of $G_{\mathcal{L}}$ and combine layer-wise the features of nodes. We thus obtain an embedding under the network schema view for each target entity i, as defined in Eq. 3.8:

$$\mathbf{z}_i^{\mathrm{NS}} = \sum_{l\in L} \beta^{(l)} \mathbf{z}_{\langle i,l\rangle}^{\mathrm{NS}}, \tag{3.8}$$

where $\beta^{(l)}$ is the learned attention coefficient for layer G_l, computed via the same attention model like in Eq. 3.7, where in this case the learnable weights are shared by all layers.

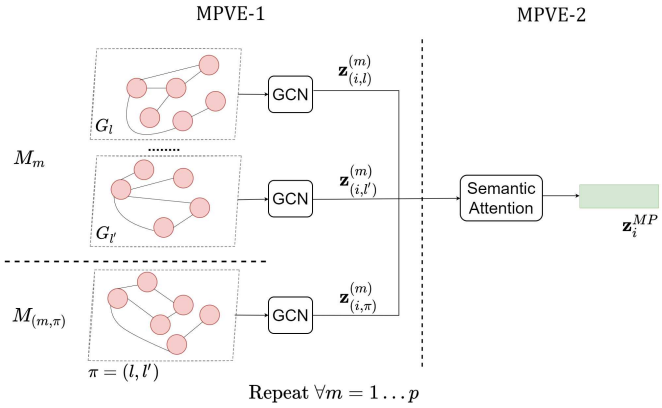


Fig. 3.3: Illustration of the sub-steps of the Co-MLHAN embedding generation of entity *i* under the meta-path view.

3.2.2.2 Meta-path view embedding

In the meta-path view, the embedding of each target node is computed from its meta-path based neighbors and from the pillar-edges derived by the corresponding meta-path based graph. We remind that each layer of a meta-path based graph is a homogeneous network with nodes corresponding to a subset of target nodes and edges as connections of meta-path based neighbors, including across-layer information matching pillar-edges.

We consider meta-paths of any length, starting and ending with nodes of target type; indeed, information of intermediate nodes can be discarded as it is included in the network schema view. Note that considering multiple meta-paths allow us to deal with multiple semantic spaces [44], and our framework is designed to handle an arbitrary number of meta-paths. Also, in case a layer does not contain any node of target type, the layer is discarded from the resulting multilayer graph. Yet, our framework admits the worst case of $\ell - 1$ layers missing for a meta-path type.

Analogously to the network schema view, the meta-path view embedding generation consists of two main steps (Figure 3.3):

(MPVE-1) First, we aggregate information of the same type, this time intended as several instances of the same meta-path and encoded via meta-path-specific Graph Convolutional Network (GCN) [39]
, obtaining, for each target node, an embedding w.r.t. each meta-path type.

(MPVE-2) Second, we combine information of different types (i.e., different meta-paths in different layers) and layers (i.e., different meta-paths across layers) via *semantic* attention, learning the importance of each meta-path and obtaining an embedding for each target node and entity under the mata-path view.

In the following, we first describe the process of meta-path view embedding generation according to the basic Co-MLHAN approach. Next, in Section 3.2.2.3,

we shall describe an alternative strategy, called Co-MLHAN-SA, which differs from Co-MLHAN in the way across-layer information relating to pillar-edges is modeled.

Aggregating information of different instances of the same type (MPVE-1).

The first step of embedding generation under the meta-path view is to aggregate information of the same type, which corresponds to several instances of a given meta-path. More specifically, we consider all p meta-paths $\mathcal{M} = \{\sigma_{m1}, \sigma_{m2}, \ldots, \sigma_p\}$ involving nodes of target type, where each meta-path σ_m matches a multilayer graph with at most ℓ layers.

In the meta-path view, across-layer dependencies are modeled as particular types of meta-paths, i.e., *across-layer meta-paths*. As defined in Section 2.3.2, they refer to the same within-layer composite relations, with the additional constraint that the terminal nodes belong to different layers, and that the intermediate node matches a pillar-edge, i.e., it corresponds to an entity with both instances involved in the composite relation. We recall that the set of *across-layer meta-paths* is defined as $\mathcal{M}^{\Updownarrow}$, as the the union of all meta-paths of any type and defined over all layer-pairs, and that $N^{\Leftrightarrow}(\cdot)$ and $N^{\Updownarrow}(\cdot)$ indicates for each node the intra-layer and inter-layer neighborhood, respectively. Note that $N_m^{\Leftrightarrow}(i,l)$ and $N_m^{\Updownarrow}(i,l)$ identify the meta-path based neighborhood of type σ_m for node $\langle i,l\rangle$, with m referring to a within or across-layer meta-path, respectively; in particular, $N_m^{\Leftrightarrow}(i,l) \equiv N_m(i,l)$.

Given any target node $\langle i,l\rangle$, we apply a meta-path specific graph neural network f_m (with K hidden layers) in order to compute its embedding according to the m-th meta-path; formally, at each k-th layer:

$$\mathbf{z}_{\langle i,l\rangle}^{(k+1)} = \begin{cases} f_m^{(k+1)}\left(\mathbf{z}_{\langle i,l\rangle}^{(k)}, \bigoplus\{\mathbf{z}_{\langle j,l\rangle}^{(k)} \mid \langle j,l\rangle \in N_m^{\Leftrightarrow}(i,l)\}\right) & \text{if } m \text{ is within layer mp} \\ f_m^{(k+1)}\left(\mathbf{z}_{\langle i,l\rangle}^{(k)}, \bigoplus\{\mathbf{z}_{\langle j,l\rangle}^{(k)} \mid \langle j,l\rangle \in N_m^{\Updownarrow}(i,l)\}\right) & \text{if } m \text{ is across-layer mp} \end{cases} \tag{3.9}$$

where $\mathbf{z}_{\langle i,l\rangle}^{(0)} = \mathbf{h}_{\langle i,l\rangle}$ is the feature embedding computed in the first stage, and $\bigoplus$ denotes an arbitrary differentiable function, aggregating feature information from the local neighborhood of nodes (e.g., summation, a pooling operator, or even a neural network [91]). Similarly to [93], we use a GCN architecture as f_m, for all M_m ($m = 1 \ldots p$) in Eq. 3.9, assuming no different contribution from different instances of the same meta-path.

More specifically, given the m-th within-layer meta-path and $\mathcal{A} = \{\mathbf{A}_1, \cdots, \mathbf{A}_\ell\}$ as the set of adjacency matrices associated with the corresponding meta-path based graph, being $\mathbf{A}_l \in \mathbb{R}^{n_l \times n_l}$ ($l = 1..\ell$) the adjacency matrix associated with layer l, the GCN for layer G_l is defined as follows:

$$\mathbf{z}_{\langle i,l\rangle}^{(k+1)} = \sigma\left(\sum_{\langle j,l\rangle \in N^{\Leftrightarrow}(i,l)} \frac{1}{\sqrt{\widetilde{\mathbf{D}}_{ii}^{l}\widetilde{\mathbf{D}}_{jj}^{l}}} \mathbf{W}^{(k,l)^{\mathrm{T}}} \mathbf{z}_{\langle j,l\rangle}^{(k)}\right) \tag{3.10}$$

where $\sigma(\cdot)$ is a non-linear activation function (default is $ReLU(\cdot) = max(0, \cdot)$), $\mathbf{W}^{(k,l)}$ is the trainable weight matrix for the m-th meta-path in the k-th convolutional layer of shape (d, d), and $\widetilde{\mathbf{D}}^{l}_{ii} = \sum_j \widetilde{\mathbf{A}}^{l}_{ij}$ is the degree matrix derived from $\widetilde{\mathbf{A}_l} = \mathbf{A}_l + \mathbf{I}_n$, with $\mathbf{I}^{l}_{n}$ as the identity matrix of size n_l, and n_l number of nodes of layer G_l. The GCN model for across-layer meta-paths is built similarly, considering $N^{\Updownarrow}(\cdot)$ instead of $N^{\Leftrightarrow}(\cdot)$ and π instead of l.

Let $\mathbf{z}^{(m)}_{\langle i,l\rangle}$ and $\mathbf{z}^{(m)}_{\langle i,\pi\rangle}$ be the node embedding associated with the m-th within (resp. across)-layer meta-path of node $\langle i, l\rangle$ (resp. layers-pair π). Downstream of meta-path specific GNNs, we obtain $\{\mathbf{z}^{(m)}_{\langle i,l\rangle} \mid l \in L,\ m = 1 \ldots p\} \bigcup \{\mathbf{z}^{(m)}_{\langle i,\pi\rangle} \mid \langle m, \pi\rangle \in \mathcal{M}^{\Updownarrow}\}$ node embeddings.

Combining information of different types and layers (MPVE-2).

Once obtained the meta-path specific embeddings for each target node, we employ *semantic-level attention* for combining different meta-path types, including both intra- and inter-layer information. Given a node $\langle i, l\rangle$, the embedding under the meta-path view is computed as follows:

$$\mathbf{z}^{\text{MP}}_{\langle i,l\rangle} = \sum_{m=1}^{p} \beta^{(m,l)} \mathbf{z}^{(m)}_{\langle i,l\rangle} + \lambda^{\Updownarrow} \left(\sum_{m=1}^{p} \sum_{\pi \mid l \in \pi} \beta^{(m,\pi)} \mathbf{z}^{(m)}_{\langle i,\pi\rangle} \right), \tag{3.11}$$

where β is the attention coefficient denoting the importance of each type of within layer and across-layers meta-path (cf. Eq. 3.7) and $\lambda^{\Updownarrow} \in [0..1]$ is a balancing coefficient denoting the importance of inter-layer connections.

In order to project the node embedding into the same space of the loss function – analogously to the network schema view – we aggregate the embeddings obtained from each layer with a sum operator, which is defined as follows:

$$\mathbf{z}^{\text{MP}}_{i} = \sum_{l \in L} \mathbf{z}^{\text{MP}}_{\langle i,l\rangle}. \tag{3.12}$$

Note that Eq. 3.12 does not require an additional level of attention, since the layer dependency has already been taken into account by the attention mechanism in Eq. 3.11. Therefore, Eqs. 3.11 and 3.12 can be combined as follows:

$$\mathbf{z}^{\text{MP}}_{i} = \underbrace{\sum_{m=1}^{p} \sum_{l \in L} \beta^{(m,l)} \mathbf{z}^{(m)}_{\langle i,l\rangle}}_{\text{within-layer}} + \underbrace{\lambda^{\Updownarrow} \left(\sum_{m=1}^{p} \sum_{\pi \in L_{cross}} \beta^{(m,\pi)} \mathbf{z}^{(m)}_{\langle i,\pi\rangle} \right)}_{\text{across-layers}}. \tag{3.13}$$

Equation 3.13 hence enables the direct computation of the final embedding under the meta-path view for each entity i.

3.2.2.3 Alternative Meta-path view embedding: Co-MLHAN-SA

Our alternative approach for embedding generation under the meta-path view is named Co-MLHAN-SA, where the suffix 'SA' refers to the *supra-adjacency matrix* modeling each meta-path based graph. The supra-adjacency matrix, denoted as $\mathbf{A}^{\text{sup}}$, has diagonal blocks each representing a layer-specific adjacency matrix (i.e., $\mathbf{A}_l \in \mathbb{R}^{n_l \times n_l}$, with $l = 1..\ell$), and off-diagonal blocks each corresponding to the inter-layer adjacency matrix $\mathbf{A}_\pi$ for layer-pair $\pi = (l, l')$, with values equal to 1 if an edge between $\langle i, l \rangle$ and $\langle j, l' \rangle$ exists, with $l \neq l'$, and 0 otherwise.

To give an intuition, we model across-layer information downstream of semantic attention, by accounting for another level of attention, i.e., across-layer attention (by analogy with the network schema view).

We thus learn the importance of different (within layers) meta-paths via semantic attention, obtaining an embedding under the meta-path view for each node and we subsequently learn the importance of each layer via across-layer attention, obtaining an embedding under the meta-path view for each entity.

Like in the basic Co-MLHAN approach, the meta-path view embedding generation in Co-MLHAN-SA consists of two main steps (Figure 3.4):

(MPVE-SA-1) First, we aggregate information of the same type, intended as several instances of the same meta-path and encoded via meta-path-specific GCNs, obtaining, for each node, an embedding w.r.t. each meta-path. Unlike MPVE-1, the first step of the Co-MLHAN-SA approach hence handles the inter-layer dependencies derived from pillar-edges.

(MPVE-SA-2) Second, we combine information of different types (i.e., different meta-paths in different layers) via *semantic* attention, learning the importance of each meta-path and obtaining an embedding for each target node under the meta-path view. Moreover, we combine information from different layers via *across-layer* attention, learning the importance of each layer and obtaining, for each target entity, a single embedding under the meta-path view.

By avoiding across-layer meta-paths $\mathcal{M}^{\Updownarrow}$ definition, Co-MLHAN-SA requires a limited number of learnable parameters, as it utilizes a meta-path specific GCN shared by all layers G_l.

Aggregating information of different instances of the same type (MPVE-SA-1).

We still use the notation $N^{\Leftrightarrow}(\cdot)$ and $N^{\Updownarrow}(\cdot)$ to indicate the set of within-layer and across-layer neighbors, respectively. As noted in Section 2.3.2, while the definition of $N^{\Leftrightarrow}(\cdot)$ does not change w.r.t. the basic Co-MLHAN, the definition of $N^{\Updownarrow}(\cdot)$ of the Co-MLHAN-SA approach is modified in the modeling of pillar-edges, by directly considering all the instances of the same target entities in other layers.
Similarly to MPVE-1, we apply a meta-path specific GNN for aggregating different meta-path instances of the same type:

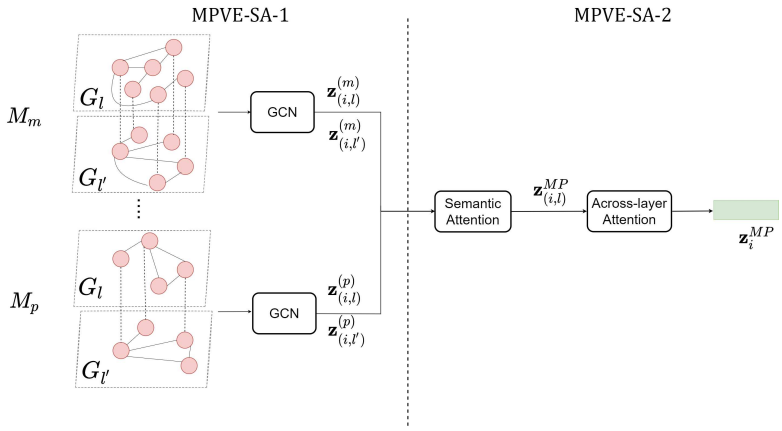


Fig. 3.4: Illustration of the sub-steps of the Co-MLHAN-SA embedding generation of entity i under the meta-path view.

$$\mathbf{z}^{(k+1)}_{\langle i,l\rangle} = f^{(k+1)}_m \left(\mathbf{z}^{(k)}_{\langle i,l\rangle}, \bigoplus^{k} \left(\{ \mathbf{h}^{(k)}_{\langle j,l\rangle} \mid \langle j,l\rangle \in N^{\Leftrightarrow}(i,l) \cup N^{\Updownarrow}(i,l) \} \right) \right). \tag{3.14}$$

Unlike MPVE-1, the inter-layer dependencies are taken into account by the GNN, employing a modified version of the propagation rule that can handle the supra-adjacency matrix as input. We thus build for each meta-path its corresponding *meta-path based supra-graph*, i.e., a graph where pillar edges exist between every node and its counterpart in other coupled layers. In our setting, we instantiate f_m with a multi-layer GCN model [107], as shown in Eq. 3.15:

$$\mathbf{z}^{(k+1)}_{\langle i,l\rangle} = \sigma \left(\sum_{\langle j,l'\rangle \in N^{\Leftrightarrow}(i,l) \cup N^{\Updownarrow}(i,l)} \frac{1}{\sqrt{\widetilde{\mathbf{D}}_{ii}\widetilde{\mathbf{D}}_{jj}}} \mathbf{W}^{(k,m)^{\mathrm{T}}} \delta(l,l')\, \mathbf{z}^{(k)}_{(j,l')} \right), \tag{3.15}$$

where the degree matrix $\widetilde{\mathbf{D}}$ is built considering both inter-layer and intra-layer links of nodes using the supra-adjacency matrix of the graph, $\widetilde{\mathbf{D}}_{ii} = \sum_{j=1} \widetilde{\mathbf{A}}^{\text{sup}}_{ij}$, where $\widetilde{\mathbf{A}}^{\text{sup}}$ is the supra-adjacency matrix with self-loops added, $\delta(l,l')$ is a scoring function denoting the weight coefficient for inter-layer links, ranging between 0 and 1, with values equal to $\lambda^{\Updownarrow}$ if $l \neq l'$, and 1 otherwise.

Let $\mathbf{z}^{m}_{\langle i,l\rangle}$ be the embedding of node $\langle i,l\rangle$ associated with the m-th metapath. We thus obtain $\bigcup_{\substack{m=1\ldots p \\ l\in L}} \{\mathbf{z}^{(m)}_{\langle i,l\rangle}\}$ meta-path specific embeddings.

Combining information of different types and layers (MPVE-SA-2).

Once obtained the meta-path specific embeddings for each target node, we employ *semantic-level attention* for combining different meta-path types, obtaining for each node $\langle i, l \rangle$ an embedding under the meta-path view, which is defined as follows:

$$\mathbf{z}^{\text{MP}}_{\langle i,l \rangle} = \sum_{m=1 \ldots p} \beta^{(m,l)} \mathbf{z}^{(m)}_{\langle i,l \rangle}, \tag{3.16}$$

where $\beta^{(m,l)}$ is an attention coefficient computed as in Eq. 3.7.

In order to project the node embedding into the same space of the loss function, we apply an additional level of attention, named *across-layer attention*, similarly to network schema view, thus obtaining for each entity i an embedding under the meta-path view:

$$\mathbf{z}^{\text{MP}}_{i} = \sum_{l \in L} \beta^{(l)} \mathbf{z}^{\text{MP}}_{\langle i,l \rangle}, \tag{3.17}$$

where $\beta^{(l)}$ is the attention coefficient denoting the importance of the l-th layer, computed similarly to Eq. 3.7.

3.2.3 Final embedding based on contrastive learning

The third stage of the proposed framework is concerned with the exploitation of a contrastive learning mechanism to produce the final entity embeddings, pulling together similar entities and pushing apart dissimilar ones in the embedding space. We combine the contrastive losses computed according to each view, with individual nodes of both positive and negative pairs selected from distinct views.

Given the embeddings $\mathbf{z}^{\text{NS}}_i$ (Eq. 3.8) and $\mathbf{z}^{\text{MP}}_i$ (either Eq. 3.13 or Eq. 3.17) for each target entity i, we transform them into the same space in which a contrastive loss function is computed, by employing a simple MLP architecture with one hidden layer, as defined in Eq. 3.18:

$$\begin{aligned} \hat{\mathbf{z}}^{\text{NS}}_i &= \mathbf{W}^{(2)} \sigma(\mathbf{W}^{(1)} \mathbf{z}^{\text{NS}}_i + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}, \\ \hat{\mathbf{z}}^{\text{MP}}_i &= \mathbf{W}^{(2)} \sigma(\mathbf{W}^{(1)} \mathbf{z}^{\text{MP}}_i + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}, \end{aligned} \tag{3.18}$$

where $\mathbf{W}^{(2)}$, $\mathbf{W}^{(1)}$, $\mathbf{b}^{(2)}$ and $\mathbf{b}^{(1)}$ are learnable weights shared by both views and $\sigma(\cdot)$ is the activation function (default is ELU).

The contrastive loss according to a certain view is computed on pairs of positive and negative samples. While earlier contrastive learning approaches were based on one or more negatives and a single positive for each instance, we follow the more recent trend of using both multiple positive and negative pairs [37, 93]. Each target entity i can hence rely on more than one positive (at least itself, under the other view). For positive sampling, the idea is to select the best nodes connected by multiple meta-path instances, since meta-path based neighbors have higher probability of

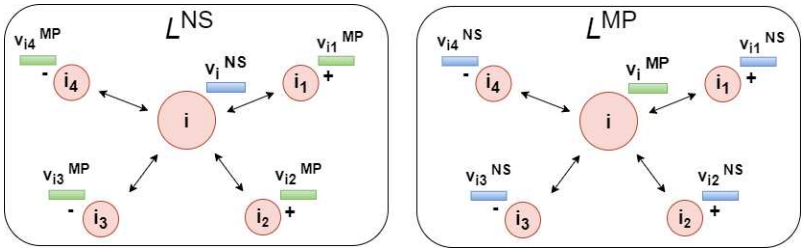


Fig. 3.5: Illustration of multi-view contrastive learning. For L^{NS}, the embedding of the target entity i is under the network schema view (colored in blue), while positive (+) and negative (-) samples are under the meta-path view (colored in green). By contrast, for L^{MP}, the embedding of the target entity i is under the meta-path view (colored in green), while positive (+) and negative (-) samples are under the network schema view (colored in blue).

being similar to each other. For negative sampling, we simply choose considering everything that is not positive.

We first proceed to the selection of positive samples. For this purpose, we count the meta-paths instances connecting each pair of target entities, considering all meta-path types on individual layers, as shown in Eq. 3.19:

$$C_{i,j} = \sum_{l \in L} \sum_{m=1\ldots p} |\{j \mid \langle j, l \rangle \in N_m^{\Leftrightarrow}(i, l)\}|. \tag{3.19}$$

For each target entity i, we obtain a set $\mathcal{S}_i = \{j \in \mathcal{V} \mid C_{i,j} > 0\}$ which is sorted by decreasing values of $C_{i,j}$. Given a threshold T_{pos}, we select for each entity itself and the best $T_{pos} - 1$ entities as positives, obtaining a subset $\overline{\mathcal{S}}_i \subseteq \mathcal{S}_i$ with $|\overline{\mathcal{S}}_i| \leq T_{pos} - 1$; all the remaining $|\mathcal{V}| - T_{pos}$ entities are regarded as negatives for i. Therefore, for each entity i, we define the set of **positive samples** $\mathcal{P}_i$ as $\mathcal{P}_i = i \cup \{j | j \in \overline{\mathcal{S}}_i\}$ and the set of **negative samples** $\mathcal{N}_i$ as $\mathcal{N}_i = \mathcal{V} \setminus \mathcal{P}_i$.

We stress that for the selection of positives we only exploit structural information, without using any information derived from the encoding of external content (i.e., initial features) of entities. Nonetheless, additional conditions on meta-paths in the selection of entity pairs can be defined, e.g., by diversifying the minimum number of instances required to enable the enumeration of a specific meta-path. Co-MLHAN is flexible in both the meta-path counting method and the overall positive and negative selection strategy.

For the computation of contrastive losses according to a given view, the embedding of each target entity i is selected from the given view, while the positive and negative samples are selected from the other view, as defined in Eqs. 3.20 and 3.21, and illustrated in Figure 3.5:

$$L^{\mathrm{NS}} = -\log \frac{\sum_{j \in \mathcal{P}_i} \exp(sim(\hat{\mathbf{z}}_i^{\mathrm{NS}}, \hat{\mathbf{z}}_j^{\mathrm{MP}})/\tau)}{\sum_{u \in \mathcal{P}_i \cup \mathcal{N}_i} \exp(sim(\hat{\mathbf{z}}_i^{\mathrm{NS}}, \hat{\mathbf{z}}_u^{\mathrm{MP}})/\tau)}, \tag{3.20}$$

$$L^{\mathrm{MP}} = -\log \frac{\sum_{j \in \mathcal{P}_i} \exp(sim(\hat{\mathbf{z}}_i^{\mathrm{MP}}, \hat{\mathbf{z}}_j^{\mathrm{NS}})/\tau)}{\sum_{u \in \mathcal{P}_i \cup \mathcal{N}_i} \exp(sim(\hat{\mathbf{z}}_i^{\mathrm{MP}}, \hat{\mathbf{z}}_u^{\mathrm{NS}})/\tau)}, \tag{3.21}$$

where $sim(\mathbf{v}_1, \mathbf{v}_2)$ denotes the cosine similarity between two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, and τ is the *temperature* parameter, which indicates how concentrated the embeddings are in the representation space, so that a lower temperature leads the loss to be dominated by smaller distances and widely separated representations contribute less. Note that Eqs. 3.20-3.21 are independent from the specific strategy of positive and negative selection; we leave the investigation of alternative sampling methods as future work (Sect. 3.4).

The final contrastive loss is computed as a convex combination of the two contrastive losses to balance the effects of the two views:

$$L_{co} = \lambda L^{\mathrm{NS}} + (1 - \lambda) L^{\mathrm{MP}} \tag{3.22}$$

with $0 < \lambda < 1$. The loss function is completely specified depending on whether an unsupervised or semi-supervised paradigm is adopted. The extension to the (semi-)supervised case can be done by adding a new term to the final loss, as shown in Eq. 3.23:

$$L_{tot} = \eta L_{co} + L_{sup} \tag{3.23}$$

where L_{sup} is the (semi-)supervised term, e.g., cross-entropy for classification tasks, jointly optimized with the contrastive term in a end-to-end fashion, and the coefficient η, $0 \leq \eta \leq 1$, is given to the contrastive term, since in a (semi-)supervised setting the (semi-)supervised term is expected to be more relevant.

Similarly to [8], once the training procedure is completed, the optimized $\mathbf{z}_i^{\mathrm{MP}}$ or $\mathbf{z}_i^{\mathrm{NS}}$ will eventually be used for downstream tasks. Particularly, our default choice is to select the embeddings under the meta-path view, since meta-paths represent high-order relations between target nodes and pillar edges capture the information of instances of the same entity, exploiting multilayer dependencies. It should however be noted that the similarity between the two learned embeddings, for any entity, is expected to be high, since, according to our positive selection strategy, each entity i includes itself under the other view in its set of positive samples $\mathcal{P}_i$. Nonetheless, in Sect. 3.3.3, we shall provide empirical evidence of such embedding similarities. The final learned embeddings optimized via such cross-view contrastive loss can be used for a wide range of analysis tasks — at node, entity, or edge level — such as node/entity classification, graph clustering, link prediction.

3.3 Experimental evaluation

In this section, we describe the experimental evaluation of our framework. Our main goal is to evaluate Co-MLHAN and Co-MLHAN-SA on the entity (multi-class) classification task, choosing a target node type among the different node types with replicas in multiple layers and real-world initial features both at node and entity-level. Section 3.3.1 introduces the data, Section 3.3.2 presents the competing methods, Section 3.3.3 discusses the experimental settings, and Section 3.3.4 describes the main results.

3.3.1 Data

To the best of our knowledge, there is a lack in the literature of publicly available benchmarks/repositories of networks that are simultaneously multilayer, heterogeneous, and attributed. To overcome this issue so as to properly build suitable network data for our evaluation, we resorted to online resources that would fulfill minimal requirements in terms of publicly availability, domain accessibility, and variety and richness of stored information. In this respect, we ended up to select the Internet Movie Database (IMDb),[2] the most popular and authoritative online resource for movies, TVs and celebrities. Note that IMDb was used in existing studies (e.g., [92, 19, 110]) for the same classification task (based on movie genres) we address in this work; however, the variety of the resulting datasets makes it hard to perform a fair comparison, beyond being incomplete in terms of our requirements (i.e., networks that are both multilayer and heterogeneous at each layer).

We constructed two IMDb network datasets, dubbed *IMDb-MLH* and *IMDb-MLH-mb* (where suffix 'mb' stands for 'most balanced'). They both model each of the layers of the multilayer network as heterogeneous (and attributed). We identify three types of entities, inherited by nodes: MOVIE (for short, M) ACTOR (for short, A) and DITECTOR (for short, D). Type MOVIE is regarded as the *target type*, therefore the downstream task is multi-class classification on movie genres, which are 'action', 'comedy' and 'drama'. Tables 3.2–3.4 summarize main characteristics of the networks, which are described next.

- *IMDb-MLH*. Our main network dataset was conceived primarily for comparative evaluation with the competitors. As it can be noticed from Table 3.4, the network is particularly unbalanced w.r.t. the distribution of classes (i.e., movie genres), which reflects a major requirement of one of our competitors, that is, to ensure that the neighbors of each node cover *all* node types. To fulfill this requirement, we hence had to select from the original dataset nodes of type MOVIE with at least one neighbor of type DIRECTOR (in any layer) and at least one neighbor of type ACTOR (in any layer), while respecting the neighborhood constraint in the monoplex, flattened network. Note that IMDb also contains MOVIE nodes with

[2] https://www.imdb.com/interfaces/

Table 3.2: Summary of within-layer network statistics.

		IMDb-MLH		*IMDb-MLH-mb*	
		2020	**2021**	**2020**	**2021**
nodes	MOVIE	1852	1459	1992	1580
	ACTOR	9165	7364	10044	8096
	DIRECTOR	3937	3003	3964	3019
edges	MOVIE -ACTOR	10966	8741	12203	9751
	MOVIE -DIRECTOR	4972	3838	5002	3860
meta-paths	MOVIE -ACTOR -MOVIE	3862	3096	5346	4414
	MOVIE -DIRECTOR -MOVIE	1874	1310	1884	1328

Table 3.3: Summary of across-layer network statistics.

		IMDb-MLH	*IMDb-MLH-mb*
entities	MOVIE	2807	3033
	ACTOR	14720	15987
	DIRECTOR	5736	5775
pillar edges	MOVIE	504	539
	ACTORr	1809	2153
	DIRECTOR	1204	1208
meta-paths	MOVIE -ACTOR -MOVIE	3417	4629
	MOVIE -DIRECTOR -MOVIE	1697	1701

Table 3.4: Distribution of the classes (i.e., movie genres) for *IMDb-MLH* and *IMDb-MLH-mb*.

	IMDb-MLH			*IMDb-MLH-mb*		
	Action	**Comedy**	**Drama**	**Action**	**Comedy**	**Drama**
no. of entities	320	1268	1219	546	1268	1219
percentages	11.4%	45.2%	43.4%	18%	41.8%	40.2%

no links with DIRECTOR or ACTOR nodes, which is however manageable by our methods only. We also filtered out MOVIES with no episode associated with a plot (plots in IMDb are entered by users, and hence it might happen that all episodes of a certain TV series are not associated with plots; or, if available, the plots could be poorly meaningful).

- *IMDb-MLH-mb*. This network dataset differs from the other one as it aims to reduce class imbalance. To this purpose, we kept the same number of 'comedy' and 'drama' MOVIE nodes as in *IMDb-MLH* and increased those of the 'action' class, by relaxing the constraint of having at least one neighbor ACTOR and one neighbor DIRECTOR for each MOVIE. Due to this relaxation, we could not use *IMDb-MLH-mb* for evaluating the competitors, but we exploited the network to further delve into our methods.

In the following, we provide a detailed description of the semantics of the constituting elements and the steps involved for data preprocessing.

For the sake of simplicity, we model a temporal network with two layers corresponding to years 2020 and 2021 of movie release. Each of the layers is modeled as heterogeneous (and attributed). Each node type, i.e., MOVIE (M), ACTOR (A) and DIRECTOR (D), can be associated with its own initial features. For instance, a MOVIE can be associated with a rating, one or more genres, film's gross and budget spent, a poster, a trailer, etc., while an ACTOR or a DIRECTOR can be associated with personal data, such as short biography, photo, a list of the most famous interpreted or directed characters, etc. An entity of type MOVIE matches a *tvSeries*, while a node of type MOVIE matches a specific *season* in a certain year. Each *season* is intended as an aggregation of *episodes*, i.e., their combined information. Pillar edges between nodes of type MOVIE refer to seasons of the same TV series in different years. As we previously stated, MOVIE is regarded as the *target type*, therefore the classification task is to predict the movie genre, i.e., 'action', 'comedy' and 'drama'.

An entity of type ACTOR or DIRECTOR matches a specific person in that role. Its corresponding nodes are included in specific layers if he/she worked in the related year. Pillar edges between nodes of type ACTOR refer to the same ACTOR who acted in some movies in different years; analogously, pillar edges between nodes of type DIRECTOR refer to the same DIRECTOR who directed some MOVIES in different years. Pillar edges are here considered as the only inter-layer relations, although our framework is designed to model non-pillar edges as well, connecting nodes possibly of different type, in different layers (e.g., MOVIES referencing other MOVIES or ACTORS referencing MOVIES).

Intra-layer edges involving nodes of target type are only between nodes of different types, and in particular between nodes of types M and A (M-A meaning "interpreted by" and A-M meaning "starred in") and between nodes of types M and D (M-D meaning "directed by" and D-M meaning "directed"). Our framework would also allow direct edges between nodes of the same type; for instance, any two MOVIES sharing a certain feature (e.g., "same genre as", "same running time as", "same original language as", etc.) can be connected. We stress that intra-layer edges in different layers are generally seen as different relation types; for instance, if different layers are built according to movie genres, a relation between the same two types of nodes in one layer can assume a different meaning in the other layers.

We select six meta-paths, two for each type of entity: M-A-M (MOVIE -ACTOR -MOVIE) and M-D-M (MOVIE -DIRECTOR -MOVIE) for type M, from which we derive pairs of MOVIES starring the same ACTOR or directed by the same DIRECTOR, respectively; A-M-A (ACTOR -MOVIE -ACTOR) and A-M-D-M-A (ACTOR -MOVIE -DIRECTOR -MOVIE -ACTOR) for type A, indicating pairs of ACTORS who acted in the same MOVIE or who acted in different MOVIES but directed by the same DIRECTOR, respectively; D-M-D (DIRECTOR -MOVIE -DIRECTOR) and D-M-A-M-D (DIRECTOR -MOVIE -ACTOR -MOVIE -DIRECTOR) for type D, identifying pairs of DIRECTORS who co-directed the same MOVIE or pairs of DIRECTORS who directed different MOVIES but with a common ACTOR, respectively. Meta-paths M-A-M and M-D-M, involving the target type, are used in the corresponding view and are both employed in meta-path count for the selection of positive samples. Specifically, for each entity pair, **AL1A** (at least 1 ACTOR) increases the meta-path count for each meta-path M-D-M or M-A-M instance

connecting the two entities, requiring at least one M-D-M or one M-A-M, i.e., the two MOVIES have at least a DIRECTOR or an ACTOR in common; **AL3A** (at least 3 ACTORS) increases the meta-path count for each M-D-M or M-A-M instance connecting the two entities, requiring at least one M-D-M or three M-A-Ms, i.e., the two MOVIES have at least a DIRECTOR or more than three ACTORS in common. As a result, *AL1A* can rely on more positives per entity but less meaningful — including MOVIE pairs sharing only one ACTOR — while *AL3A* can rely on less but more meaningful positives per entity. Main statistics of the two alternatives are provided in Table 3.5.

The across-layers meta-paths are built upon the same meta-path types, with the intermediate node matching a pillar-edge. For instance, as shown in Figure 2.7, given a meta-path of type M-A-M (for each layer), the corresponding across-layer meta-path has the same *ACTOR* in both layers and the two *MOVIES* belonging to different layers.

We provide nodes/entities of target type with real-world initial features; for the other two types, we identify initial features associating each node with an one-hot indicator vector [87]. Initial features of MOVIE nodes/entities are extracted from plots of individual episodes, where terms are selected according to their *term-frequency inverse-document frequency* (*tf-idf*) relevance scores. Specifically, we filter out words that appear in less than 10 documents or in more than 60% of the total corpus size. After that, in our experimental settings, we either selected the top-1000 words according to their tf-idf scores, or kept all (unfiltered) words (4085).

We emphasized that Co-MLHAN is conceived to be general and flexible, so as to exploit all available information but also being effective even when such information is lacking, e.g., in case of poor across-layer relationships, or when one or more types of neighbors are missing for some nodes; for instance, a new TV series could have a single season or the information regarding its cast could miss. In addition, nodes could show high variability in the number of neighbors, e.g., TV series can be associated with a large cast or not. External information can indeed be available either at node level or entity level, therefore initial features can be layer-dependent and associated with nodes, or layer-independent and associated with entities. For instance, we might handle the plots of the TV series (entities), which we also assign to the respective seasons (nodes) in different years (layers), as well as the plots of the individual seasons, from which we derive the overall plots of the series.

3.3.2 Competing methods

We compared Co-MLHAN and Co-MLHAN-SA with two unsupervised learning methods, HeCo [93] and NSHE [110], on *IMDb-MLH*. HeCo is a contrastive multi-view learning based method for single-layer heterogeneous attributed graphs. We equipped HeCo with the same meta-paths and the same positives and negatives as used by our methods. NSHE is a unsupervised non-contrastive GNN-based approach for single-layer heterogeneous attributed graphs, which is designed to learn embeddings preserving both pairwise and network schema structure. In contrast to our

Table 3.5: Summary of positive sampling statistics.

	IMDb-MLH-mb	*IMDb-MLH*	
	AL3A	**AL1A**	**AL3A**
average number of positives per entity	1.664	2.166	1.657
self positive only	2018	1550	2192
minimum number of positives per entity	1	1	1
$\mathbf{T_{pos}}$	5	5	5

methods, NSHE generates initial features of nodes by using DeepWalk [68] for all types of nodes and, if available, combines them with real-world features.

As a motivation behind our choice of competing methods, we note that HeCo and NSHE are those sharing more aspects with our methods (cf. Section 2.4). Indeed, they are able to encode local and global node structure separately in an unsupervised manner, thus capturing the heterogeneity of both nodes and relations. Moreover, they respect the network schema of the graph, ensuring to visit all types of nodes and edges, they can deal with imbalance in the number of neighbors and relations of a certain type within the network schema, and allow to focus on the generation of embeddings of a specific type while using heterogeneous information.

It should however be emphasized that HeCo and NSHE were designed for *heterogeneous attributed monoplex* networks, i.e. single-layer graphs. Consequently, we were forced to downgrade our network data through a flattening approach, i.e., by compressing the multi-layer graph into a single graph discarding all replicated edges.

3.3.3 Experimental settings

To model each of our network datasets, intra-layer edges involving nodes of target type (i.e., MOVIE) were considered between nodes of different types only, and pillar edges were considered as the only inter-layer relations, although our framework is designed to model non-pillar edges as well. Meta-paths with both terminal nodes of target type were used in the corresponding view and employed in meta-path count for the selection of positive samples. For the positive (and negative) selection strategy, we defined two alternatives, named **AL3A** and **AL1A**, differing in whether or not they consider constraints on the minimum number of instances of a specific meta-path type (*AL* stands for 'At Least'). This reflects on a different trade-off between the number of positives, which is higher in *AL1A*, and their meaningfulness, which is expected to be higher in *AL3A*. The positive statistics corresponding to the two strategies are provided in Table 3.5.

For all methods, we first learned the embedding for each entity in an unsupervised fashion and then trained a classifier for the final class prediction. We remind that for the final classification task we use the embeddings learned under the meta-path view, since it captures relations between target nodes, although our positive selection strat-

egy and the joint optimization of the loss function entail similar representations. To validate our hypothesis, for each entity i, we computed the cosine similarity between the embedding under the network schema view ($\mathbf{z}_i^{\text{NS}}$) and the embedding under the meta-path view ($\mathbf{z}_i^{\text{MP}}$). Results on *IMDb-MLH* confirmed our hypothesis, since we obtained the following statistics on the distribution of similarity measurements: 0.84 as 25% percentile, 0.87 as mean, 0.88 as median, 0.92 as 75% percentile, and 0.97 as maximum value.

We found the optimal hyperparameters for the representation learning process via grid search algorithm. Specifically, we trained the model using the Adam optimization algorithm [38] with full batch size, for 10000 epochs, with *early stopping* technique based on the contrastive loss value and patience set to 30 (i.e., the training procedure stops if loss value does not decrease for 30 consecutive epochs), with $\lambda = 0.5$ for the convex combination of the two contrastive losses. Learning rate was set to 0.0001, and dropout regularization technique with $p = 0.3$ was applied to the transformed features $\mathbf{h}$.

We used $Q = 1$ attention heads, since GATv2 showed to work better than multi-head GAT, and temperature value $\tau = 0.5$. Moreover, we set the hidden dimension (d) for both views to 64, with $K = 1$ hidden layers in the *meta-path view* (including multiple layers can often lead to over-smoothing problem [42]). In the network schema view, for neighborhood sampling, we randomly sampled 7 and 2 nodes of type ACTOR and DIRECTOR, resp., at each epoch with replacement strategy. In the meta-path view, following [93], we set the threshold for positive selection T_{pos} equals to 5. Finally, we set $\lambda^{\Updownarrow} = 1$ for the inter-layer edges, in order to fully exploit the inter-layer connections represented by pillar-edges. In case of Co-MLHAN, this setting of $\lambda^{\Updownarrow} = 1$ to give the maximum importance to the inter-layer edges, is justified by the construction of across-layer meta-paths, since their intermediate node correspond to a pillar edge (between nodes of type ACTOR or DIRECTOR). In case of Co-MLHAN-SA, we directly had pillar-edges between nodes of type MOVIE, as this proved to be effective in other works, e.g., [107].

As mentioned before, HeCo and NSHE were trained over the flattened networks, i.e., by discarding multilayer information, since they are conceived for single-layer heterogeneous graphs. While for HeCo we kept the same settings as for Co-MLHAN (cf. Section 3.3.2), for NSHE we selected the same hyperparameters it uses for the IMDb dataset [110]. For a fair comparison, we set its embedding dimension to 64. We use the publicly available software implementations for both competitors.[3]

Once obtained the final embedding, we used a MLP with one hidden layer of size 64 as final classifier, trained using the Adam optimization algorithm with full batch size, for either 2000 epochs, or at convergence when the early-stopping regularization technique was selected (with patience value of 300 epochs); in the latter case, since the macro average treats all classes equally, we used F1 score with macro average as quality criteria on the validation set, in order to penalize wrong predictions of the most unbalanced class, i.e., 'action'. We split each dataset in training, test and validation sets, by choosing 70%, 15% and 15% of the entities for each class,

[3] The HeCo and NSHE source code are publicly available at https://github.com/liun-online/HeCo and https://github.com/AndyJZhao/NSHE, respectively.

respectively. Note that, when early stopping was not used, we just discarded the validation set so as not to vary the training and test sets. The learning rate was set to 0.01.

We carried out our methods and HeCo for 5 independent runs, which differed in random seed assignment, while we experimented NSHE for one run, due to its computational overhead, thus finally learning 5 and 1 different model weights, respectively. For each trained model, we derived the final network embeddings — to be given as input to the final classifier — and executed the final classifier over 50 independent runs with the same realization of training, test and validation sets. Finally, we computed the average of the performance scores achieved on the test set. Specifically, for each model, we computed *F1-score* with *micro* and *macro averaging*, *AUC score*, and F1-score of each class. F1-score with micro and macro averaging is used to evaluate the contributions of all classes, considering individual class contributions or treating all classes equally, respectively. ROC AUC (Area Under the Receiver Operating Characteristic Curve) score with OVR (one-vs-rest) averaging strategy is used to indicate the ability of the classifier to distinguish between classes. We also report F1-score for each class to more effectively evaluate how the model performances are affected by the early stopping technique. Note that for methods from which multiple models were learned (i.e., they were executed over different seeds), we reported the average values for each performance criterion.

3.3.4 Results

We organize the presentation of our experimental results into four parts: Sections 3.3.4.1 and 3.3.4.2 concern the evaluation on *IMDb-MLH* and *IMDb-MLH-mb*, respectively, whereas Section 3.3.4.3 provides a qualitative analysis of the learned embeddings. Finally, Section 3.3.4.5 summarizes our experimental findings.

3.3.4.1 Evaluation on *IMDb-MLH*

We first compared Co-MLHAN and Co-MLHAN-SA with HeCo and NSHE using initial features corresponding to the best top-1000 words by tf-idf and positives selection under the tougher condition *AL3A*, which assumes fewer but higher-quality positives per node (cf. Appendix B); moreover, to ensure a fair comparison with our competitors requiring a flattening approach, we used for our methods only features associated with entities (entity-level features, for short *EL*).

We tested the classifier both with and without early-stopping technique. In both cases, as shown in Table 3.6, our proposed methods achieve high performance scores according to all quality criteria, consistently outperforming the competitors. In fact, although the amount of edges that were "lost" due to the flattening approach is relatively small (15 % and 20%, resp.), the compression of all layers does not allow the competitors to suitably capture the relations on different layers as well as their

Table 3.6: Results on *IMDb-MLH* (1000 features), with and without early-stopping. (Bold values refer to the best score for each criterion)

Method	Early-stopping	F1 micro	F1 macro	AUC	F1 'action'	F1 'comedy'	F1 'drama'
Co-MLHAN	No	**0.7174 ± 0.0158**	**0.6406 ± 0.0513**	**0.8017 ± 0.0225**	0.4452 ± 0.1402	0.7375 ± 0.0124	**0.7390 ± 0.0133**
Co-MLHAN-SA	No	0.6980 ± 0.0143	0.6336 ± 0.0444	0.8004 ± 0.0193	**0.4672 ± 0.1233**	0.7176 ± 0.0126	0.7159 ± 0.0129
HeCo	No	0.5250 ± 0.0068	0.3595 ± 0.0107	0.5880 ± 0.0112	0.0083 ± 0.02540	0.6118 ± 0.0169	0.4583 ± 0.0190
NSHE	No	0.5778 ± 0.0146	0.4900 ± 0.0198	0.6673 ± 0.0145	0.2587 ± 0.0493	0.5963 ± 0.0168	0.6150 ± 0.0167
Co-MLHAN	Yes	0.7141 ± 0.0159	0.6151 ± 0.0723	0.7975 ± 0.0238	0.3682 ± 0.1987	**0.7400 ± 0.0127**	0.7370 ± 0.0158
Co-MLHAN-SA	Yes	0.6969 ± 0.0153	0.6089 ± 0.0493	0.7958 ± 0.0204	0.3849 ± 0.1340	0.7242 ± 0.0114	0.7175 ± 0.0142
HeCo	Yes	0.5287 ± 0.0056	0.3626 ± 0.0093	0.5903 ± 0.0097	0.0056 ± 0.0227	0.6150 ± 0.0055	0.4671 ± 0.0081
NSHE	Yes	0.5821 ± 0.0139	0.4886 ± 0.0493	0.6857 ± 0.0108	0.2450 ± 0.0331	0.5954 ± 0.0169	0.6253 ± 0.0164

inter-layer dependencies. Note that we could not apply our competitors on a single layer of our network, since many entities are missing in each layer; as shown in Table 3.3, only 504 out of 2807 target entities are shared between the two layers.

Co-MLHAN achieves the best performances on almost all the quality criteria (5 out of 6), while Co-MLHAN-SA, being the approach with closer performance, is the most effective in predicting movies of class 'action' (0.467), which is the less represented class. The reason behind this slight difference between our two methods might be due to the different across-layer information modeling w.r.t. pillar-edges. The across-layer meta-paths defined by Co-MLHAN can be more meaningful, as they exploit richer inter-layer information than Co-MLHAN-SA. Moreover, the poor performance of HeCo w.r.t NSHE show that the contrastive learning mechanism performed by HeCo is not very effective for this dataset. Particularly, HeCo shows the lowest performance on the 'action' class, indicating that its learned embedding is unable to discriminate the instances of the most unbalanced class.

Impact of early-stopping on the entity classification. Focusing on the results obtained by using the early-stopping technique, the overall performance of our methods turns out to be slightly lower than the setting discarding the early-stopping. In particular, from Table 3.6, we notice that the F1-score values corresponding to the 'action' class decrease for all methods when the early-stopping technique is used. We indeed found out that in some runs the training procedure stops too early because the F1 macro computed on validation set does not improve within the patience value. In this respect, Figure 3.6 shows the testing and validation F1 macro scores of the final classifier averaged over 50 runs of the same (i.e., fixed-seed) model of Co-MLHAN, with and without early-stopping technique. When choosing early-stopping, the best-performing epochs are distributed with a mean value of 234 ± 272, while the 25%, 50% (median) and 75% percentiles are equal to 14, 32 and 421, respectively. Since the increase in the F1 macro occurs around the 400th epoch (Fig. 3.6, left), the classifier appears to be under-trained in some runs, thus it cannot boost its performance. On the other hand, if the training is not early-stopped, the classifier learns to distinguish more accurately the instances of the most unbalanced class in each run.

The above results would suggest that, in the effort of avoiding overfitting and saving computational resources through the early-stopping technique, the final classifier might be under-trained, leading to an underfitting problem if the patience value is not properly set. In fact, we observed that the F1 macro on the validation set stabilizes

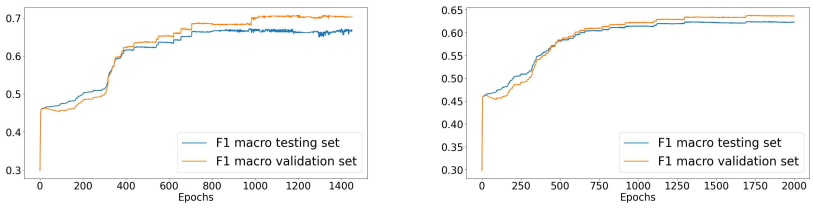


Fig. 3.6: Testing and validation F1 macro for the final classifier with early-stopping (left) and without early-stopping (right).

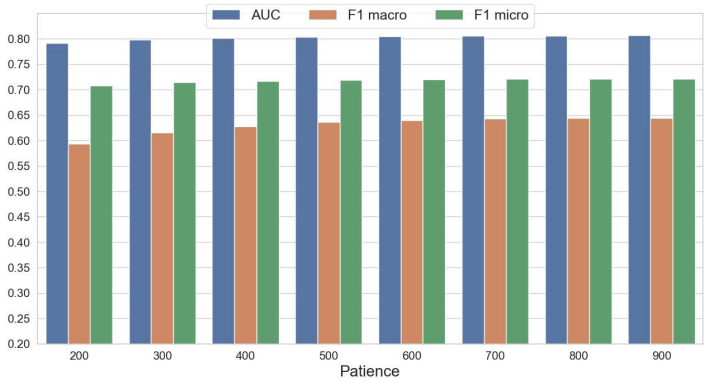


Fig. 3.7: Quality criteria of the final classifier with different patience values. The best performance is reached with patience equal to 900: 0.807 AUC, 0.645 F1 macro, 0.721 F1 micro.

around the 1000-th epoch (Fig. 3.6, right); however, as shown in Figure 3.7, the overall benefit gained by a high patience value is marginal: a patience value set to 900 led to 0.644 F1 macro, which just decreases to 0.615 if the patience is set to 300, with only an improvement on the most unbalanced class, as shown in Figure 3.8.

We point out that the hyperparameters of the final classifier were not globally optimized, since this goes beyond the main focus of this work; nonetheless, we recall that the classifier is shared by our methods and the competing ones, so as to fulfill fairness in the comparative evaluation. We therefore preferred to speed up the classification stage and set the patience value to 300 for all the experiments employing early-stopping technique on the classifier.

Impact of initial feature selection. We analyzed the behavior of the methods when equipped with all initial real features, i.e., without constraining the size of the initial feature space. We carried out the experiments with the same positives selection strategy as in the previous evaluation. Results corresponding to the early-stopping

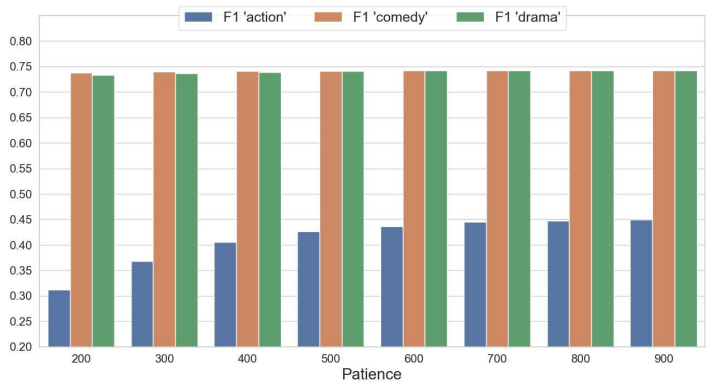


Fig. 3.8: Quality criteria on each class of the final classifier with early-stopping technique with different patience values.

Table 3.7: Results on *IMDb-MLH* (4000 features), with early stopping. (Bold values refer to the best score for each criterion)

Method	F1 micro	F1 macro	AUC	F1 'action'	F1 'comedy'	F1 'drama'
Co-MLHAN	**0.6801 ± 0.0111**	**0.6308 ± 0.0386**	**0.7968 ± 0.0154**	**0.5017 ± 0.1106**	**0.6986 ± 0.0093**	**0.6922 ± 0.0104**
Co-MLHAN-SA	0.6555 ± 0.0149	0.6124 ± 0.0407	0.7788 ± 0.0207	0.4996 ± 0.1132	0.6652 ± 0.0142	0.6724 ± 0.0132
HeCo	0.5053 ± 0.0044	0.3447 ± 0.0060	0.5682 ± 0.0082	0.2598 ± 0.0232	0.6106 ± 0.0144	0.6569 ± 0.0139
NSHE	0.6052 ± 0.0120	0.5091 ± 0.0115	0.7020 ± 0.0106	0.0014 ± 0.0097	0.5931 ± 0.0046	0.4397 ± 0.0081

setting are reported in Table 3.7 (note that we observed no particular differences when not using early-stopping).

Compared to the previous case corresponding to the top-1000 initial features, the performance of all methods tends to decrease due to the higher and sparser dimensionality. An exception is represented by NSHE, which slightly improves, probably due to its feature initialization [110]. However, Co-MLHAN and Co-MLHAN-SA still outperform both competitors, with the former achieving the highest F1 micro, F1 macro and AUC values. Moreover, when keeping all words as initial features, our methods report high values on the 'action' class (despite the use of the early-stopping technique), while the competitors maintain similar values to the previous case with top-1000 initial features.

The above results hence suggest that dealing with a full space of initial features can enable Co-MLHAN and Co-MLHAN-SA to better distinguish the MOVIE instances, and in particular that our methods can effectively exploit these features unlike the competitors.

Table 3.8: Results on *IMDb-MLH-mb* (1000 features) and positives selection *AL3A*. (Bold values refer to the best score for each criterion)

Method	Early-stopping	F1 micro	F1 macro	AUC
Co-MLHAN (EL)	No	0.7401 ± 0.0089	0.7475 ± 0.0084	0.8676 ± 0.0052
Co-MLHAN-SA (EL)	No	0.7411 ± 0.0096	0.7552 ± 0.0089	0.8740 ± 0.0058
Co-MLHAN (NL)	No	**0.8810 ± 0.0071**	**0.8755 ± 0.0070**	**0.9566 ± 0.0032**
Co-MLHAN-SA (NL)	No	0.8705 ± 0.0063	0.8692 ± 0.0063	0.9475 ± 0.0034
Co-MLHAN (EL)	Yes	0.7443 ± 0.0057	0.7509 ± 0.0058	0.8769 ± 0.0029
Co-MLHAN-SA (EL)	Yes	0.7471 ± 0.0071	0.7580 ± 0.0070	0.8845 ± 0.0047
Co-MLHAN (NL)	Yes	0.8707 ± 0.0126	0.8661 ± 0.0113	0.9532 ± 0.0069
Co-MLHAN-SA (NL)	Yes	0.8694 ± 0.0059	0.8672 ± 0.0060	0.9542 ± 0.0021

3.3.4.2 Evaluation on *IMDb-MLH-mb*

We further evaluated Co-MLHAN and Co-MLHAN-SA using the *IMDb-MLH-mb* network. More specifically, we investigated the behavior of our methods when equipped with node-level initial features, hereinafter referred to as *NL*, i.e., with layer-dependent initial features. To this purpose, we first compared the methods under the following setup: initial features corresponding to the top-1000 words, positives selection *AL3A*, with and without using early-stopping technique.

As it can be noticed from Table 3.8, performance generally increases w.r.t. the entity-level feature initialized methods, especially in terms of F1 macro, as a direct consequence of a better coverage of the 'action' class. Comparing the results obtained with entity-level (*EL*) and node-level (*NL*) features, we observe that, as expected, exploiting initial features at each layer (i.e., node-level case) leads to higher performance of the methods.

Moreover, we observe that the difference between the case with early-stopping and the case without early-stopping decreases on *IMDb-MLH-mb*, regardless of the layer dependency of the initial features, i.e., *EL* or *NL* setting.

Furthermore, we changed the meta-paths count strategy for positive selection (*AL1A*) (refer to Table 3.5 and Appendix B for additional details) to test the sensitivity of our methods, without changing the feature initialization. Results shown in Table 3.9 reveal a marginal decrease in performance, slightly more evident when using node-level initial features. This might be due since, according to *AL1A*, each entity has a number of positive samples which is on average greater than for the *AL3A* alternative, but the positives can be less meaningful (cf. Appendix B); nonetheless, we observed a negligible worsening in the performance.

3.3.4.3 Qualitative inspection of the embeddings

After discussing the results from a numerical point of view, in this section we aim to visually analyze the final entity embeddings in order to gain insights in terms of patterns and clusters. To this purpose, we used Uniform Manifold Approximation

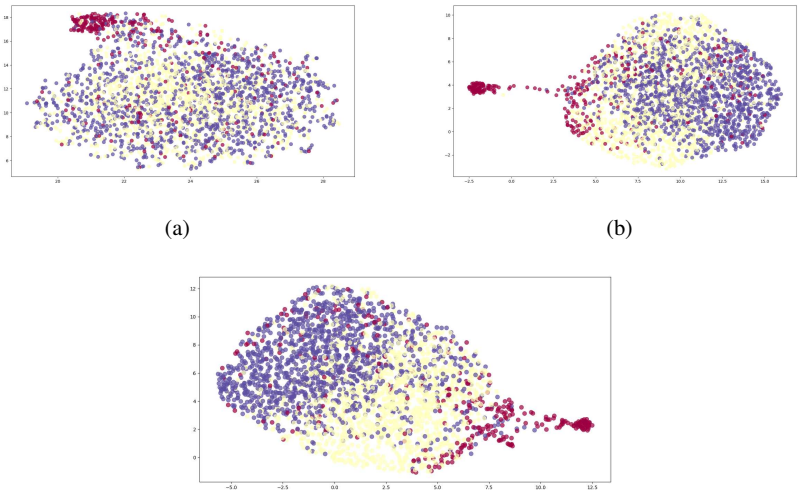
Table 3.9: Results on *IMDb-MLH-mb* (1000 features) and positives selection *AL1A*. (Bold values refer to the best score for each criterion)

Method	Early-stopping	F1 micro	F1 macro	AUC
Co-MLHAN (EL)	No	0.7387 ± 0.0092	0.7419 ± 0.0087	0.8649 ± 0.0056
Co-MLHAN-SA (EL)	No	0.7373 ± 0.0108	0.7469 ± 0.0102	0.8717 ± 0.0055
Co-MLHAN (NL)	No	0.8611 ± 0.0084	0.8588 ± 0.0081	0.9437 ± 0.0040
Co-MLHAN-SA (NL)	No	0.8676 ± 0.0067	**0.8658 ± 0.0065**	0.9449 ± 0.0036
Co-MLHAN (EL)	Yes	0.7442 ± 0.0070	0.7480 ± 0.0070	0.8681 ± 0.0051
Co-MLHAN-SA (EL)	Yes	0.7449 ± 0.0075	0.7529 ± 0.0076	0.8808 ± 0.0042
Co-MLHAN (NL)	Yes	0.8482 ± 0.0138	0.8478 ± 0.0130	0.9389 ± 0.0066
Co-MLHAN-SA (NL)	Yes	**0.8678 ± 0.0066**	**0.8658 ± 0.0072**	**0.9517 ± 0.0031**

and Projection (UMAP) [58], which is a highly effective non-linear dimensionality reduction algorithm, particularly useful for visualizing relative proximities in high-dimensional data. It is based on manifold learning, which can be seen as a generalization of linear projection frameworks like PCA, sensitive to non-linear structures in data. In recent years, UMAP has gained popularity since it offers several advantages over related algorithms, such as PCA and t-SNE [50]. In particular, compared to the latter, UMAP can achieve a better preservation of the global structure of data in the final projection, it is more efficient, and it has no computational restrictions on the embedding dimension. UMAP defines two main hyperparameters to control the balance between local and global structure: *nearest neighbors* and *minimum distance*, denoting the number of local nearest neighbors to process, and how tightly UMAP packs points together, respectively. On the one hand, lower values of minimum distance result in more clustered embeddings, while larger values prevent UMAP from packing points together, leading to a more uniform dispersion of points; on the other hand, lower values of nearest neighbors allow UMAP to concentrate more on the local structure, while higher values enable looking at more neighbors for each point, resulting in a more global representation.

Figure 3.9 shows the two-dimensional UMAP visualization of the initial feature embeddings with tf-idf weighting (Fig. 3.9a), and of the final embeddings under the meta-path view learned by our methods (Figs. 3.9b and 3.9c), w.r.t. *IMDb-MLH*. We executed UMAP with the following main hyperparameters: size of local neighborhood used for manifold approximation equal to 15, minimum distance between points equal to 0.7, and cosine similarity as proximity measure.

In the initial representation (Fig. 3.9a), all entities of type MOVIE are grouped closely together regardless of their genre, resulting in a cluttered representation. This is actually not surprising, since their plots are provided by users without meeting quality requirements. Nonetheless, Figs. 3.9b and 3.9c show how the final embeddings learned by our methods allow UMAP to better separate entities of different classes.



(a) (b)

(c)

Fig. 3.9: UMAP 2D visualization of the entity embeddings on the *IMDb-MLH* dataset, with red, yellow and purple points indicating movies of genre 'action', 'comedy' and 'drama', respectively: the initial feature embedding with tf-idf weighting (a), the final entity embedding learned by Co-MLHAN-SA (b), and the final entity embedding learned by Co-MLHAN (c).

3.3.4.4 Computational complexity aspects

In this section, we discuss the computational complexity aspects of our framework. In our analysis, we assume sparse graphs in both views, dense content-features obtained after the content encoding stage and the worst case in terms of magnitude of the networks. That is, each entity appears in each layer, i.e, the total number of nodes in the network schema view is $O(|\mathcal{V}|\ell)$ and each target node appears in the meta-path based graphs, i.e., the total number of nodes in the meta-path view is $O(|\mathcal{V}^{(t)}|\ell)$ for each meta-path. Without loss of generality, we consider that each relation $r \in R$ involves nodes of target type (thus ensuring that $|R|$ relations are considered in the network schema view), and we discard the across-layer meta-paths. Before delving into the details, we recall that the input and output of each sub-module of stage 2 and 3 are $d-$dimensional embeddings, with $d \ll |V_{\mathcal{L}}|$.

As concerns the spatial complexity, the memory requirement is mainly given by the storage of the hidden states (e.g., $\mathbf{z}^{\text{NS}}$ and $\mathbf{z}^{\text{MP}}$), the learnable weight matrices ($\mathbf{W}$s) and attention vectors ($\mathbf{a}$s). In particular, the attention values in NSVE-1 require an overhead of $|E_r|$ for each relation r involving the target nodes. Moreover, we need to store in memory the positive and the negative samples for each entity, i.e., $\mathcal{P}_i$ and $\mathcal{N}_i$, where $|\mathcal{P}_i \cup \mathcal{N}_i| = |\mathcal{V}^{(t)}|$.

Regarding the time complexity, the graph structure encoding stage requires the computation of embeddings under the network schema and the meta-path view, which can be calculated independently and therefore can be parallelized. The computational complexity of the former view is shared by both Co-MLHAN and Co-MLHAN-SA, while the latter view requires a separate analysis for the two methods. In the following, we analyze the costs of each of the steps performed at the two views.

(NSVE-1) The computational cost of the NSVE-1 step, where node-level attention takes place, depends on an attention mechanism for each relation in each layer. Given a relation type $r \in R$, let V_r be the set of nodes connected through the edges in E_r. The computational complexity of Eq. 2.32 with a single attention head is $T_r = O(|V_r|d^2 + |E_r|d)$ [4], where the first term concerns the feature transformation step of GATv2, while the second term corresponds to the cost of calculating a general attention function, which can be parallelized. In the case of Q attention heads, both the first and the second terms are multiplied by a factor of Q, where the different heads can still be parallelized. Note that in practice, each target node considers only a subset of neighbors for each relation r due to our sampling strategy, which allows saving computational resources. Hence $|E_r|$ is an upper bound to the number of edges involved in relation r. Finally, since we equipped our approaches with the same attention mechanism on each relation r, the final time complexity of NSVE-1 is $O(max(T_{r_1}, T_{r_2}, \ldots, T_{r_{|R|}}))$.

(NSVE-2) NSVE-2 employs the same multilayer perceptron model for type-level and across-layer attention. In particular, under the assumption that each relation $r \in R$ involves nodes of target type, the time complexity of the type-level attention step is $O(|\mathcal{V}^{(t)}|d^3|R|)$, because involves dense matrix and vector operations. For the across-layer attention case, under the initial hypothesis that each target entity appears in each layer, the time complexity is $O(|\mathcal{V}^{(t)}|\ell d^3)$. Also, note that in both cases the attention coefficients can be calculated in parallel, for each relation r, and layer l, respectively.

(MPVE-1) For each meta-path and layer, the complexity of MPVE-1 corresponds to the complexity of GCN [39], whose cost for K neural layers is $O(Knonzero(\mathbf{A}_l)d + K|\mathcal{V}^{(t)}|d^2)$, where $nonzero(\mathbf{A}_l)$ is the number of non-zero entries in the adjacency matrix of the l-th layer. Note that, in practical applications, K assumes small values due to the issue of oversmoothing [42], and the computations on each layer, meta-path (and across layer meta-path) are independent to each other, hence they can be easily parallelized.

(MPVE-2) MPVE-2 requires an attention model to compute the importance of each meta-path in each layer. Since the attention mechanism is the same as used in NSVE-2, the cost of MPVE-2 is $O(p|\mathcal{V}^{(t)}|\ell d^3)$, where the attention coefficients for each meta-path can be computed in parallel.

(MPVE-SA-1) Regarding Co-MLHAN-SA, the time complexity of MPVE-SA-1 corresponds to the application of ML-GCN [107] with K neural layers. Its computational complexity is $O(Knonzero(\mathbf{A}^{\text{sup}})d + K|\mathcal{V}^{(t)}|\ell d^2)$, where $nonzero(\mathbf{A}^{\text{sup}})$ is the number of non-zero entries in the $\mathbf{A}^{\text{sup}}$ matrix. The first term corresponds to the

propagation steps, while the second corresponds to the feature transformation steps of ML-GCN.

(MPVE-SA-2) Similarly to MPVE-2, this sub-module requires the application of semantic-level attention, in order to combine the embedding learned from each multilayer meta-path based graph. Since we discarded across-layer meta-paths in MPVE-2, the computational complexity of this step is the same for both Co-MLHAN and Co-MLHAN-SA, i.e., $O(p|\mathcal{V}^{(t)}|\ell d^3)$. Also, similarly to NSVE-2, MPVE-SA-2 requires to attend over the information learned at each layer, with a level of across-layer attention, whose complexity is negligible compared to the first term, i.e., $O(|\mathcal{V}^{(t)}|\ell d^3)$.

The third stage, based on contrastive learning, requires first a transformation through a MLP, which costs $O(|\mathcal{V}^{(t)}|d^2)$, then the loss functions of the two views are computed. For this last step, we need to compute the pairwise cosine-similarities between nodes belonging to different views, which costs $O(|\mathcal{V}^{(t)}|^2 d)$.

To sum up, considering all the above terms, the time complexity of our framework can be characterized in terms of size of the multilayer heterogeneous network and size of the latent space (i.e., embedding length), which is typical in GNN-based approaches. Specifically, in the second stage, the cost is linear in the number of target nodes and edges, while it is cubic in the embedding length, due to the computation of the attention models. In the third stage, the cost becomes quadratic in the number of the target entities, due to the calculation of pairwise node similarities. We remark that our framework is extremely flexible in terms of the choice of each sub-module. In particular, we propose using an attention mechanism only if different instances of the same type are assumed to provide information with different importance. Nonetheless, several steps can be carried out in parallel (e.g., the attention model on each relation r, GCN models for each meta-path, type-level, semantic-level and across-layer attention). Thus, in practical applications, the computational complexity of our framework does not hinder its scalability. In this regard, we aim to improve the efficiency of the training process in future works, e.g., by equipping it with mini-batch training setting [24], or investigating more efficient similarity methods.

3.3.4.5 Summary of results

In this section we summarize the main findings of the empirical evaluation of our framework. We experimented it on two novel network datasets derived from IMDb (cf. Appendix B), which are simultaneously multilayer, heterogeneous, and attributed. Specifically, we modeled IMDb as a temporal network with two layers, where each layer is heterogeneous and corresponds to years of movie releases. The first network dataset, named *IMDb-MLH*, was conceived for the comparative evaluation of our framework, since it fulfills the requirements of our competitors. The second network dataset, named *IMDb-MLH-mb*, was designed to reduce class imbalance and is not applicable to the competitors. Thus, we used it to investigate different input settings of our methods, i.e., Co-MLHAN and Co-MLHAN-SA.

Experimental results on the entity classification task showed that our methods significantly outperform existing competitors, effectively exploiting both external content and multilayer information. We also demonstrated that the overall performances do not degrade even in the (less realistic) case of feature-set size greater than the number of target nodes. In this case, our methods obtained higher values on the most unbalanced class, suggesting that Co-MLHAN and Co-MLHAN-SA can effectively exploit the full space of initial features. To ensure fairness in the evaluation, the final MLP classifier was shared by all methods. Moreover, we investigated the impact of early-stopping regularization technique on the final classifier, confirming that underfitting phenomena can arise if the patience value is not properly set.

We further inspected the quality of the learned embeddings through a data visualization tool, showing that our cross-view contrastive mechanism is beneficial for the downstream classification task, since instances belonging to different genres are properly clustered w.r.t. the initial embedding with only tf-idf information. As a related aspect, we provided evidence that, as theoretically expected, the embeddings under the meta-path view share a similar structure with the corresponding embeddings under the network-schema view, thus enabling the use for downstream tasks of the embeddings learned under one or the other view.

We investigated further properties of our methods using *IMDb-MLH-mb*. In that stage of evaluation, the difference between the case with and without early-stopping is strongly mitigated by the lower imbalance between classes. We showed that our framework is resilient to the selection of positive samples (*AL1A* vs. *AL3A*), and able to effectively exploit node-tailored feature information (*NL* vs. *EL*).

It should also be noted that our Co-MLHAN and Co-MLHAN-SA, which differ in the meta-path view, achieved similar performance in all the experiments, showing that both approaches can successfully handle information coming from pillar edges. Specifically, the performance by Co-MLHAN would suggest that defining meta-paths between different layers (i.e., across-layer meta-paths) allows one to suitably integrate high-order relations between nodes in different layers.

3.4 Concluding remarks

In this work, we proposed a self-supervised graph representation learning framework, based on a cross-view contrastive learning mechanism, for networks that are simultaneously multilayer, heterogeneous and attributed. Remarkably, our framework is able to deal with networks where each layer is a heterogeneous graph with attributed nodes, and with both intra- and inter-layer links between nodes. The embedding of nodes of any given target type are learned by contrasting the encodings generated by two views, i.e., network schema view and meta-path view, which embed local and high-order neighborhood information, respectively. The meta-path view also encodes across-layer information. To this purpose, two variants of the framework are proposed: Co-MLHAN, modeling a particular type of meta-paths with terminal nodes belonging to different layers and the intermediate node matching a pillar-edge,

and Co-MLHAN-SA, directly considering all the instances of the same target entities in other layers. The learned embeddings are task-independent and hence can eventually be used for different downstream graph mining tasks, both at entity/node level, edge level or graph level. We demonstrated our methods under a task of entity classification, based on originally developed network datasets in the IMDb movie context, and including a comparative evaluation with recently proposed methods for heterogeneous graph embedding, HeCo and NSHE.

Possible extensions and future directions. Although our framework can deal with an arbitrary number of layers, this reflects on the number of learnable parameters, thus impacting on the framework complexity. Particularly, for Co-MLHAN, the number of learnable parameters increases with the number of layers in both views, while Co-MLHAN-SA is less sensitive to the number of layers in the meta-path view, but is still affected in the network-schema view, since we distinguish relations of the same type across different layers. To reduce the number of learnable parameters of the framework, one direction would be to modify the network schema view so as to make node-level attention weights for a certain relation type be shared over all layers. As discussed in Section 3.3.4.4, the computational complexity of our framework does not hinder its scalability, since several steps can be easily parallelized. We leave as future work the training of the models based on a mini-batch setting in combination with sampling methods [24, 7, 108, 27].

Another aspect that might be addressed concerns the modeling of meta-paths connecting nodes of different types, where at least one (rather than both) among the starting and ending node is of target type. In this case, the resulting meta-path based graph would not be homogeneous, since the meta-path based neighbors are of different types. Since increasing the number of views is unlikely to be beneficial (as stated in [25]), the definition of the two views should hence be revised.

A further extension would concern the definition of different selection strategies for the positive and/or the negative samples in the contrastive learning stage. On the one hand, the learned features could be exploited for the positives selection in addition to structural information, and on the other hand, hard negative sampling techniques could be devised [34, 1, 73].

Our framework can also be extended to deal with different graph mining tasks other than node/entity classification, such as regression, clustering, link prediction. For instance, to accomplish the latter, we would need to handle the embeddings downstream of one of the two views at node-level so as to compute pair-wise hidden representations of nodes (upon which a similarity function can be used to predict the link strength of any pair of nodes). Equally interesting would be to investigate other applications of our framework in different scenarios, having different structural and semantic properties, stressing the flexibility of the proposed framework by identifying datasets with more or less overlap between layers, and possibly with one or more node types without replicas. Contextually, by identifying richer sources of information, we could inspect other learning paradigms, such as *multi-modal* or *multi-task* learning, where multiple tasks are solved simultaneously, which has been proven effective for the task of recommendation in heterogeneous networks [43].

Chapter 4
Deep graph representation learning for dynamic, heterogeneous, attributed networks

With real-world network systems typically comprising a large number of interactive components and inherently dynamic, Graph Continual Learning (GCL) has gained increasing popularity in recent years. Furthermore, most applications involve multiple entities and relationships with associated attributes, which has led to widely adopting Heterogeneous Information Networks (HINs) for capturing such rich structural and semantic meaning. In this context, we deal with the problem of learning multi-type node representations in a time evolving graph setting, harnessing the expressive power of Graph Neural Networks (GNNs). To this purpose, we propose a novel framework, named DyHANE - Dynamic Heterogeneous Attributed Network Embedding, which dynamically identifies a representative sample of multi-typed nodes as training set and updates the parameters of a GNN module, enabling the generation of up-to-date representations for all nodes in the network [52].

We show the advantage of employing HINs on a data-incremental classification task. We compare the results obtained by DyHANE on a multi-step, incremental heterogeneous GAT model trained on a sample of changed and unchanged nodes, with the results obtained by either the same model trained from scratch or the same model trained solely on changed nodes. We demonstrate the effectiveness of the proposed approach in facing two major related challenges: (i) to avoid model re-train from scratch if only a subset of the network has been changed and (ii) to mitigate the risk of losing established patterns if the new nodes exhibit unseen properties. To the best of our knowledge, this is the first work that deals with the task of (deep) graph continual learning on HINs.

4.1 Introduction

Contemporary scenarios involve a wide variety of actors and relationships that evolve over time, making networks suitable models for analyzing data interdependence. In a dynamic discrete-time setting with changes in network structure and entities features, detecting and adapting to changes is crucial for an accurate model, so

that it can generate effective representations for all nodes in the network. It is necessary to integrate new knowledge, by identifying new patterns in the data, while simultaneously refining existing knowledge, by consolidating existing patterns. This approach helps avoiding the so-called *catastrophic forgetting problem*, i.e., the tendency of a neural network to lose proficiency in previously learned tasks when adapting to or acquiring new information [57].

As noted in [36], Graph Neural Networks (GNNs) offer refined graph representations and flexibility in handling attributes, but scalability issues limit their efficiency compared to shallow approaches. For this reason, there are surprisingly few GNN-based works on Graph Continual Learning (GCL), i.e., continual learning approaches capable of harnessing the expressive power of GNNs for the integration of new knowledge in graph representations without relying on the entire network. Even less explored is the study of *incremental* GNNs for HINs, i.e., GNNs trained incrementally to simultaneously detect new patterns in heterogeneous nodes and consolidate current ones. We specify that, although in the literature the terms 'incremental' and 'streaming' are often used interchangeably, in the following we will exclusively employ the former to emphasize the incremental, step-wise nature of the proposed approach.

Existing GNN-based works on heterogeneous networks need to store the history of nodes and apply an additional recurrent architecture or attention mechanism to update the node representations, as shown in [104, 102, 55]. In contrast, multiple architectures for homogeneous graphs, i.e., graphs with only one type of node and relationship, have been proposed to overcome scalability issues arising from storing multiple embeddings for each node or retraining the network from scratch at each observation time [48, 111, 89, 67].

The authors in [89] outline three scenarios for handling evolving graph data by employing GNN-based methods: (i) *pre-trained GNNs*, employing a pre-trained GNN model for generating representations of unseen and changed nodes, although effectiveness might decrease if node patterns significantly differ from the pre-trained model; (ii) *retrained GNNs*, which train a new GNN module on the entire graph data at each timestamp, achieving higher performance, but incurring substantial time and space costs, particularly if only a subset of the network changes over time; (iii) *online GNNs*, learning node representations by training solely on new nodes using parameters from the previous timestamp, despite a risk of loss of knowledge and degraded representations of unchanged nodes if patterns in new nodes differ from the existing network. All these strategies have clear limitations in handling the complexities of large-scale incremental graph data characterized by shifting pattern distributions over time.

To strike a balance between learning new knowledge and preserving existing knowledge, several works employ a regularization technique, such as the Elastic Weight Consolidation (EWC) [41], by introducing a regularization term in the loss function that penalizes large changes in important weights learned during previous timestamps. The regularization term effectively constrains the neural network from making drastic changes to the parameters that are crucial for the performance on unchanged nodes.

A recent research direction involves replay-based (or reharsal) methods storing representative history data or well-designed data representations, and incorporating these elements during the training of new tasks. Strategies like *experience replay* and *generative replay* based on the storing, respectively, of a representative subset of past data [111, 89, 67] or of synthetic samples resembling data [90] in a memory buffer, proved to be effective in preventing the loss of knowledge in homogeneous graphs, but rehearsal approaches are still unexplored in HINs.

To fill the above gap in the literature, we propose a novel framework, named **DyHANE** (**Dy**namic **H**eterogeneous **A**ttributed **N**etwork **E**mbedding), which is designed to generate at each new timestamp multi-typed up-to-date node representations for all nodes in the network, by incrementally updating GNN parameters training on subset of the network. To this aim, we identify a novel strategy to detect and integrate the set of nodes affected by changes in a HIN — new knowledge detection — and a novel strategy to update the heterogeneous memory buffer used for experience node replay — existing knowledge consolidation. . We experimentally evaluate our framework on a multi-class node classification task on a closed-world setting, i.e., known and fixed number of classes, handling both changes in network topology and node features. First, we demonstrate the advantage of HIN modeling. Then, we assess the effectiveness of DyHANE in comparison to Retrained GNNs and Online GNNs in terms of achieved performance and training time. We notice the flexibility of the proposed framework, particularly our base model, namely a Graph Attention Network, is actually interchangeable with any other GNN model.

Plan of the chapter. The remainder of this chapter is structured as follows. Section 4.2 describes our proposed framework in detail and also provides a discussion on computational complexity aspects. Section 4.3 provides our experimental evaluation, which was carried out by referring to the task of classification of authors' primary field of study as a case in point. Finally, Section 4.4 contains concluding remarks and provides pointers for future research.

4.2 Proposed framework

Our proposed DyHANE is an incremental GNN model. Given a dynamic attributed HIN $G^t = \langle \mathcal{V}^t, \mathcal{E}^t, A, R, \phi, \varphi, \mathcal{X}^t \rangle$ at a generic timestamp t, and assuming a discrete set of timestamps $\{t_1, t_2, ..., t_T\}$, each of which corresponds to a set of events ${\mathcal{E}_c}^t$ determining a change in the network topology $\Delta G^t = G^t - G^{t-1}$ and/or in the attribute matrices $\Delta \mathcal{X}^t = \mathcal{X}^t - \mathcal{X}^{t-1}$ (cf. Section 2.3.3), an incremental GNN gradually learn $\{\theta^{t_1}, \theta^{t_2}, ..., \theta^{t_T}\}$, where θ^t are the GNN parameters at timestamp t able to generate effective representations ${z_i}^t \forall i \in \mathcal{V}^t$ using only a sample of nodes in G^t as training set. The training set $\mathcal{T}^t$ comprises all the unseen and changed nodes plus a curated subset of unchanged nodes, i.e., $\mathcal{T}^t = \mathcal{I}^t \bigcup \mathcal{B}^{t-1}$, where $\mathcal{I}^t \supseteq \Delta \mathcal{V}^t$, $\mathcal{B}^{t-1} \subset \mathcal{V}^{t-1}$ and $\mathcal{I}^t \bigcup \mathcal{B}^{t-1} \subseteq \mathcal{V}^t$. $\mathcal{I}^t$ denotes the influenced node set at time t, i.e., a reduced subset of nodes affected by changes; $\mathcal{B}^{t-1}$ denotes the experience memory buffer

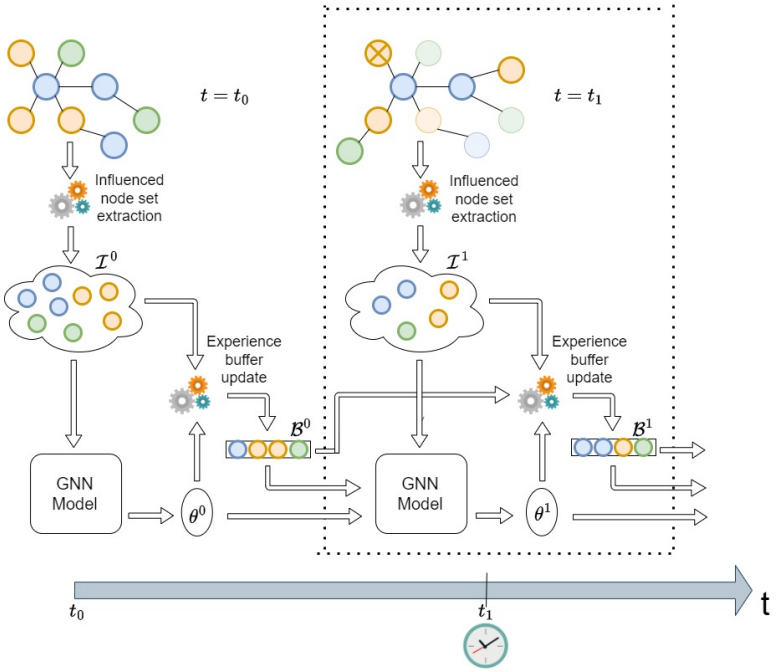


Fig. 4.1: Workflow of DyHANE, with the dashed rectangle highlighting a given incremental step $t = t_1$, involving the detection of the influenced node set at current time $\mathcal{I}^1$ (cf. Algorithm 2) and the resume of the experience buffer $\mathcal{B}^0$ and GNN initialization parameters θ^0 updated at previous timestamp. The model outputs the updated parameters of the neural network model θ^1, which should be able to generate effective representations for both changed and unchanged nodes in $\mathcal{V}^1$. Finally, the memory buffer $\mathcal{B}^1$ is updated for the next interval (cf. Algorithm 3).

updated at time $t-1$ and replayed at time t. Both $\mathcal{I}^t$ and $\mathcal{B}^{t-1}$ comprise nodes of different types.

DyHANE thus consists of a sequence of steps reiterated at each new timestamp t, as summarized in Algorithm 1 and depicted in Figure 4.1:

1. Detection of the influenced node set $\mathcal{I}^t$, as presented in Algorithm 2.
2. Resume of the experience memory buffer $\mathcal{B}^{t-1}$ computed at previous timestamp and GNN parameters θ^{t-1} learned at previous timestamp.
3. Update of the GNN module, learning θ^t able to generate effective representations for all nodes $i \in \mathcal{V}^t$.
4. Update of the memory buffer $\mathcal{B}^t$ from $\mathcal{I}^t \bigcup \mathcal{B}^{t-1}$, as presented in Algorithm 3.

Note that (1) and (2) can be run in parallel, while (4) can be performed at any time while waiting for the next increment. In the following, we describe the two proposed

Algorithm 1 DyHANE incremental GNN.

Require: Set of events ${\mathcal{E}_c}^t$ at current timestamp, experience node buffer $\mathcal{B}^{t-1}$, GNN parametrized by θ^{t-1}, number of epochs num_epochs.
Ensure: GNN parametrized by θ^t learned at current timestamp, updated experience buffer $\mathcal{B}^t$.
1: Obtain influenced node set $\mathcal{I}^t$ from ${\mathcal{E}_c}^t$ via Algorithm 2.
2: **if** $t > 0$ **then**
3: Load the experience buffer $\mathcal{B}^{t-1}$ and the GNN model initialed with θ^{t-1}.
4: **else**
5: Load G^0 and initialize the GNN parameters at random.
6: **end if**
7: Build training set $\mathcal{T}^t = \mathcal{I}^t \bigcup \mathcal{B}^{t-1}$
8: Train - test - validation splitting as in [7], or load the test set for fair comparison (cf. Sec. 4.3).
9: $i = 0$
10: **while** i < num_epochs+1 **do**
11: Calculate loss function $\mathcal{L}$ (e.g., cross-entropy for node classification)
12: Update GNN parameters using SGD
13: $i = i + 1$
14: **end while**
15: Update the experience buffer $\mathcal{B}^t$ from $\mathcal{I}^t \bigcup \mathcal{B}^{t-1}$ via Algorithm 3
16: **return** θ^t

algorithms for the detection of influenced nodes and the update of the experience buffer, respectively.

Please also note that our chosen GNN model is a Graph Attention Network (GAT) [4], which is widely recognized as a highly effective GNN for various downstream tasks; however, it should be emphasized that DyHANE is not constrained to a particular GNN architecture or to a unique implementation of the proposed strategies.

4.2.1 Identification of influenced nodes

To identify nodes affected by changes, focusing on the entire network is unnecessary, especially when the node neighborhood is stable [14]. Likewise, relying solely on changed nodes is inadequate, since the new patterns can effect existing nodes due to interdependence in network data. Given this premise, we propose a dedicated procedure (Algorithm 2) to mine nodes affected by changes during network evolution.

To identify a reduced set of influenced nodes at current timestamp $\mathcal{I}^t$, we categorize new events in the graph as *strong* or *weak* based on their impact on the network topology. The intuition is that weak events do not generate new knowledge, while strong events need to be propagated further. Inspired by recent work on graph representation learning for dynamic homogeneous networks [84] and acknowledging the effectiveness of meta-path based models in capturing heterogeneous information in large networks [13, 76], we classify the edge corresponding to an event as *weak* if it is not crossed by any meta-path instance connecting target node types, or if generated meta-path instances already exist; otherwise it is said *strong*. For all events, we add to the set of influenced nodes the incident nodes. For strong events,

Algorithm 2 Influenced node set detection.

Require: Set of events ${\mathcal{E}_c}^t$ at current timestamp, set of edges $\mathcal{E}^{t-1}$, set of attribute matrices $\mathcal{X}^{\mathbf{t-1}}$ and set of meta-path-based adjacency matrices of target node type $\overline{a} \bigcup_{\sigma_m} \mathcal{A}\text{adj}_m^{t-1}$ at previous timestamp.

Ensure: Influenced node set $\mathcal{I}^t$.

Initialize $\mathcal{I}^t = \emptyset$
for all $e_{ij} \in {\mathcal{E}_c}^t$ **do**
    Update $\mathcal{X}^t_{\phi(i)}$ and $\mathcal{X}^t_{\phi(j)}$
    $\mathcal{I}^t = \mathcal{I}^t \bigcup \{i, j\}$
    Initialize event category as $c = 1$ (*strong*)
    Generate new meta-paths instances mps for each type σ_m crossing e_{ij} and connecting nodes of target type $\overline{a}$
    if $mps = \emptyset$ **then**
        $c = 0$ (*weak*)
    else
        for all pair of terminal nodes $p \in$ mps($i.e., \forall\ \sigma_m$ crossing e_{ij}) **do**
            $ok = false$
            for all meta-path-based adjacency matrix $\mathcal{A}\text{adj}_m^{t-1}$ **do**
                if $\mathcal{A}\text{adj}_m^{t-1}[p[0]][p[1]] == 0$ **then**
                    $ok = true$
                    break
                end if
            end for
            if not ok **then**
                break ($c = 1$ (*strong*))
            end if
        end for
        $c = 0$ (*weak*)
    end if
    if c == 1 **then**
        Update $\mathcal{I}^t = \mathcal{I}^t \bigcup N^t(i) \bigcup N^t(j)$
        for all $x \in \{i, j\}$ **do**
            if $\phi(x) = \overline{a}$ **then**
                Update $\mathcal{I}^t = \mathcal{I}^t \bigcup N^t_m(x) \forall \sigma_m \in \mathcal{M}$
            end if
        end for
    end if
end for
Remove duplicates: $\mathcal{I}^t$ = set($\mathcal{I}^t$)
return $\mathcal{I}^t$

we add to $\mathcal{I}^t$ also the incident nodes' heterogeneous one-hop neighborhood; if any of the incident nodes is of target type $\overline{a} \in A$, the affected nodes will include also their meta-path-based neighborhood. Since a node can contribute to multiple events, we ensure not to include duplicates in the influenced node set. Note that to handle attribute changes of isolated nodes, i.e., nodes not involved in any change on network topology, we map the corresponding event as a particular self-loop and add the node and its neighborhood to $\mathcal{I}^t$.

Because only incremental meta-paths on changed nodes are computed, and categorization is done based on look-ups of sparse matrices, the proposed algorithm has

Algorithm 3 Experience memory buffer update.

Require: The model at current timestamp t — including training set $\mathcal{T}^t = \mathcal{I}^t \bigcup \mathcal{B}^{t-1}$, ground truth y, fixed buffer size $|\mathcal{B}|$, number of nodes to be used to compute type importance $top-k$.
Ensure: Updated experience buffer $\mathcal{B}^t$.
1: Initialize the Explainer (e.g., the CaptumExplainer).
2: Initialize a vector $\mathbf{s}^t \in R^{1\times|A|}$ to store node type importance s.t. it sums to 1.
3: Compute the importance score (impact on the gradient) $\delta_i \forall (not\ masked) i \in \mathcal{I}^t \bigcup \mathcal{B}^{t-1}$ via the Explainer.
4: **for all** node type $a \in A$ **do**:
5: Compute the relative importance of features of $top-k$ nodes of type a (sum) and store the obtained score in $\mathbf{s}^t[a]$.
6: Compute the number $top-k_a$ of nodes of type a to be stored in $\mathcal{B}^t$ weighted on $\mathbf{s}^t[a]$.
7: Select the $top-k_a$ nodes of type a from $\mathcal{I}^t \bigcup \mathcal{B}^{t-1}$ with highest impact score δ_i as sum of the importance of their features, and store them in $\mathcal{B}^t$.
8: **end for**
9: **return** $\mathcal{B}^t$

been shown to be effective in practice, compared to training on the entire network. We elaborate on this aspect in Section 4.2.3.

4.2.2 Update of the memory buffer

To update the memory buffer $\mathcal{B}^t$, we detect the most relevant nodes for classification using a Captum-based explainer, able to identify crucial subgraph structures and node features influencing GNN predictions. Employing the Integrated Gradients algorithm [83] for multi-instance explanations, we assess each input feature's contribution to the model output and identify nodes with major impact on gradient (Algorithm 3).

Integrated Gradients is a principled approach that satisfies several desirable properties for feature attribution, including completeness — the sum of the attributions exactly accounts for the difference in the model's output between the actual input and the baseline — and sensitivity — small variations in the input features lead to proportionally small changes in the attributions. Specifically, the Integrated Gradients algorithm comprises several steps:

- Baseline selection: Choose a baseline or reference point as a point in the input space with known or neutral feature values.
- Path construction: Define a path from the baseline to the input data point, being a straight line in the input space.
- Gradient calculation: Compute the gradient of the model's output with respect to the input features at multiple points along the constructed path, by taking the partial derivative of the model's output with respect to each input feature.
- Integration: Integrate the computed gradients along the path, using the trapezoidal rule or a more sophisticated method. The result is a set of values, each representing the accumulated effect of a specific feature along the path.

- Attribution Calculation: Multiply the integrated gradients by the difference between the input and baseline at each point along the path and sum these values, yielding the attribution of each feature to the model's output.
- Scaling: Scale the attributions by the difference between the baseline and the actual input, helping ensure that the attributions are meaningful and consistent across different inputs.

In the process, scores are assigned to each feature of each node. We determine the node-level score by summing along the feature dimension, i.e., by aggregating the contributions of all its features. Subsequently, we compute the cumulative contributions of the $top\text{-}k$ nodes for each node type, with k being a given constant. This computation results in the determination of the percentage of nodes for each type $a \in A$ to be stored in the experience memory buffer. $\mathcal{B}^t$ will store the $top\text{-}k_a$ nodes for each node type a, i.e., the nodes of type a with the most substantial impact on the gradient. We point out that the most frequent type in the memory buffer does not necessarily correspond to the target type $\overline{a}$ and can change at each new computation.

Note that the update of the experience memory buffer to be replayed at next timestamp can be performed at any time while waiting for the new batch of events. Different buffer sizes according to different $top\text{-}k$ thresholds were investigated in Section 4.3.

4.2.3 Computational complexity aspects

In this section, we discuss the computational complexity of DyHANE. In our analysis, we assume network evolution at discrete time, i.e., a set of *deferred* timestamps $\{t_1, t_2, \ldots, t_T\}$ corresponding to as many sets of events determining changes in network topology and/or in node attributes (cf. Section 2.3.3).

We assume sparse graphs (both the initial network G^0 and each network increment ΔG^t), and dense content-features of nodes. We also make the reasonable assumption that each increment is significantly smaller with respect to the network size , i.e., $|\mathcal{E}_c^t| \ll |\mathcal{E}^t|$ and $|\mathcal{I}^t| \ll |\mathcal{V}^t|$. We recall that the size of node embeddings is equals to d, with $d \ll |\mathcal{V}^t|$, and that the training set is $\mathcal{T}^t = \mathcal{I}^t \bigcup \mathcal{B}^{t-1}$, with $|\mathcal{B}^{t-1}| \ll |\mathcal{I}^t|$.

In the following, we delve into the time computational aspects of the proposed approach, then we analyze the memory space required between two consecutive updates and for storing intermediate representations.

4.2.3.1 Time complexity

The time complexity analysis of DyHANE can be divided according to the three main steps of Algorithm 1, namely the detection of the influenced node set (Algorithm 2), the update of the GNN module, and the update of the memory buffer (Algorithm 3).

Detection of the influenced node set. Algorithm 2 requires, for each event (edge) $e \in \mathcal{E}_c^t$, the generation of all instances for each meta-path type crossing the edge and connecting nodes of target type, being the most expensive operation. Given an edge of type r, generating all instances for a certain meta-path type σ_m crossing r costs $\prod_{\hat{r} \in R_{\sigma_m - r}} |\mathcal{E}_{\hat{r}}|$, with $R_{\sigma_m - r}$ denoting all the edge types except for (the first) r crossed by the k-meta-path type. The number of products is thus equal to the length of the meta-path type minus 2. Since these computations can be carried out independently, and hence can be parallelized, the time cost is dominated by the longest meta-path type, i.e., $O(\max(len(\sigma_1), len(\sigma_2), \ldots, len(\sigma_{\mathcal{M}}))$. To summarize, the time cost of the detection of the influenced node set is $O((|\mathcal{E}_c^t| \times |\mathcal{E}_{\tilde{r}}^{t-1}|^{\text{len}(\sigma_{\tilde{m}})-2})$, with $\tilde{r} = \max_{r \in R} (|\mathcal{E}_r^{t-1}|)$ and $\sigma_{\tilde{m}}$ being the meta-path type of maximum length. For each identified meta-path instance, a lookup at the corresponding meta-path adjacency matrix is required to check if the pair of terminal nodes already exists. We assume the worst case in terms of number of computations, i.e., all possible meta-path instances generated and all node pairs checked; nonetheless, in practical scenarios, the most informative meta-paths are usually few and of short length (cf. Section 2.3.1).

Update of the GNN module. The time complexity of a GNN is typically determined by the feature transformation step, the neighborhood aggregation (message passing) step, and architectural aspects of the neural network such as the number of layers and, in a GAT, the number of attention heads. We denote with $\mathcal{E}_{\mathcal{T}^t}$ the set of edges of the subgraph induced by the nodes in $\mathcal{T}^t$. Assuming each node's attention mechanism requires computation time linear in the number of neighboring nodes, the time complexity of message passing for each node can be expressed as $O(max_{deg} \times d)$, where max_{deg} is the maximum node degree and d is the dimension of the input feature vector for each node. Considering all nodes, it results in $O(|\mathcal{T}^t| \times max_{deg} \times d)$. In the heterogeneous case, having the same attention mechanism for each relation r, the time complexity is dominated by the most abundant relationship $r^* = \max_{r \in R} (|\mathcal{E}_{\mathcal{T}_r^t}|)$. The computational complexity with a single attention head in a general GAT is $O(|V_r|d^2 + |E_r|d)$ [4], where V_r is the set of nodes connected through the edges in E_r. The first term concerns the feature transformation step of GATv2, while the second term corresponds to the cost of calculating a general attention function, which can be parallelized. In the case of Q attention heads, both the first and the second terms are multiplied by a factor of Q, where the different heads can still be parallelized. The time cost of our heterogeneous incremental GAT module is hence $O(|\mathcal{T}_{r^*}| \times d^2 + |\mathcal{E}_{\mathcal{T}_{r^*}}| \times d)$. We note that for the same GNN architecture, the number of training iterations required for convergence usually decreases when training on a smaller subset of the graph, since the model learns from fewer examples.

Update of the memory buffer. In our setting, the detection of the most relevant nodes for classification to be stored as experience replay is accomplished by the Captum-based explainer on the GNN model. The time cost of the explainer accounts for the model evaluation and the Integrated Gradients calculation. The model evaluation time is linear with respect to the size of the input graph, i.e., $O(|\mathcal{T}^t|)$. The time cost of the Integrated Gradients calculation also depends on the size of the input

graph scaled on the number of steps in the integration process ($s = 50$ by default), i.e., $O(|\mathcal{T}^t| \times s)$. This is hence the dominating term of the time cost of the explainer.

Once obtained the output score for each node, we sort the output scores independently for each node type, which is $O(|\mathcal{T}^t_{\tilde{a}}| \times \log |\mathcal{T}^t_{\tilde{a}}|)$, with $\tilde{a}^t$ denoting the nodes of type $a \in A$. This operation is hence dominated by the type with more nodes. The computation of the relative importance of the top-k nodes for a node type (with k shared by all node types and timestamps) is $O(k)$, since it is the sum of the top-k scores for that type, and is negligible given $k \ll |\mathcal{T}^t_{\tilde{a}}|, \forall a \in A$. Analogously, selecting the top-k_a nodes for each node type on the sorted scores has a time complexity of $O(k_a)$, which is negligible given $k_a \ll |\mathcal{T}^t_{\tilde{a}}| \forall a \in A$. The total time cost of the update of the memory buffer is $|\mathcal{T}^t| \times s + |\mathcal{T}^t_{\tilde{a}}| \times \log |\mathcal{T}^t_{\tilde{a}}|$

By summing the above contributions, the overall time cost of DyHANE is $O((|\mathcal{E}^t_c| \times |\mathcal{E}^{t-1}_{\tilde{r}}|^{\text{len}(\sigma_{\tilde{m}})-2}) + (|\mathcal{T}_{r^*}| \times d^2 + |\mathcal{E}_{\mathcal{T}_{r^*}}| \times d)) + (|\mathcal{T}^t| + |\mathcal{T}^t| \times s + |\mathcal{T}^t_{\tilde{a}}| \times \log |\mathcal{T}^t_{\tilde{a}}|)$, where, we recall that, $\mathcal{E}^t_c$ is the set of events (changed edges) at current time, $\mathcal{E}^{t-1}_{\tilde{r}}$ is the set of edges of G^{t-1} of type $\tilde{r}$, with $\tilde{r}$ being $\max_{r \in R}(|\mathcal{E}^{t-1}_r|)$, $\sigma_{\tilde{m}}$ is the meta-path type of maximum length, $\mathcal{T}_{r^*}$ is the set of nodes of training set connected via the edges in $\mathcal{E}_{r*}$, with r^* being $\max_{r \in R}(|\mathcal{E}_{\mathcal{T}^t_r}|)$, $\mathcal{T}^t$ is the training set at current time, and $\mathcal{T}^t_{\tilde{a}}$ is the subset of nodes of current training set of type $\tilde{a} \in A$, with $\tilde{a}$ being $\max_{a \in A}(|\mathcal{T}^t_a|)$.

It can be noticed that the total time cost of the proposed approach, at each timestamp, is dominated by the detection of nodes affected by changes (Algorithm 2) and the training of the GNN module. The memory buffer update, beside being a computationally lighter operation, can be performed downstream of classification at any time while waiting for the next increment, exploiting knowledge of the network architecture and stored parameters for subsequent model initialization.

4.2.3.2 Space complexity

As concerns the space complexity, the memory requirement is mainly given by the storage of the node attributes ($|\mathcal{V}^t| \times d$), the edges $\mathcal{E}^t$, the learned parameters θ^t — including the weights associated with the attention mechanism and any other learnable parameters in the model — and the meta-path adjacency matrices $\mathcal{A}\text{adj}^{t-1}_m$ for all meta-path types $\sigma_m \in \mathcal{M}$ connecting target node types ($|\mathcal{V}_{\bar{a}}| \times |\mathcal{V}_{\bar{a}}| \times |\mathcal{M}|$), which are stored for the next increment. A fixed storage space is devoted to the indices of the nodes included in the memory buffer.

Algorithms 2 and 3, as well as the GAT, require additional space to store intermediate representations. Algorithm 2 needs to store, for each event (edge), all meta-path instances crossing that edge. During training and inference, GAT stores intermediate representations of nodes and attention weights, whose space complexity depends on factors such as the size of the graph and the number of layers in the model (2 in our case). Algorithm 3 needs to store the GAT model as input of the Captum explainer, and subsequently the output score for each node, and the top-k indices and scores, resulting in $O(|\mathcal{T}| + \sum_{a \in A} k_a)$, where the first term is the cost for storing the output

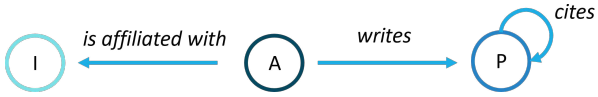


Fig. 4.2: Network schema of the proposed collaboration - citation - affiliation HIN.

scores and the second term is the cost for storing the top-k indices and scores for each node type, assuming each index and score is stored using a constant amount of memory.

4.3 Experimental evaluation

In this section, we describe the experimental evaluation of our framework. Our main goal is to assess the effectiveness of DyHANE with respect to the Retrained and Online approaches, showing that the proposed heterogeneous incremental GNN model strikes a satisfying balance between the performance achieved by Retrained GNNs, in terms of accuracy in classifying changed and unchanged nodes, and by Online GNNs, in terms of efficient incorporation of new knowledge.

Section 4.3.1 introduces the data, Section 4.3.2 discuss the advantage offered by HIN models, Section 4.3.3 presents the competing methods, Section 4.3.4 discusses the experimental settings, and Section 4.3.5 describes the main achieved results.

4.3.1 Data

Our heterogeneous dataset is a collaboration-citation-affiliation network comprising 3 different node types, i.e., Author (A), Paper (P), Institution (I), i.e., and 3×2 relation types, i.e., "A writes P" (A-P), "P cites P" (P-P), "A is affiliated with I" (A-I), and their counterparts, as shown in Figure 4.2. We selected A as target node type, towards which we built 3 different meta-paths: co-autorship (A-P-A), citation (A-P-P-A) and affiliation (A-I-A), i.e., we are interested in pairs of authors who wrote the same paper, or one cited the other, or have the same affiliation. Table 4.1 shows the dataset statistics in terms of number of nodes, number of edges and number of meta-path instances, with focus on the last increment. Table 4.2 provides more insights about centrality, connectivity, path-based and mesoscopic measures. We underline that since there are no specific libraries for computing the statistics of multi-types of nodes and relationships, we derived from the heterogeneous graph several subgraphs (bipartite, weighted, directed and undirected). For each measure we report the specific subgraph on which it is computed.

We built our dataset from OpenAlex [70] by a combination of paper attributes, including the language ('en'), the type ('journal article' or 'proceedings-articles' or

Table 4.1: Dataset statistics, in terms of no. of nodes, edges and meta-path instances, with focus on the last increment (2022).

		2019-2022	2019-2021	2022
# Nodes	Authors	15397	13116	2281
	Papers	10174	8642	1532
	Institutions	1831	1612	219
# Edges	A-P	18768	16020	2748
	A-I	4893	4181	712
	P-P	192	113	79
# Meta-paths	A-P-A (w)	65687	57736	7951
	A-I-A (w)	25567	18507	7060
	A-P-PA (w)	587	302	285

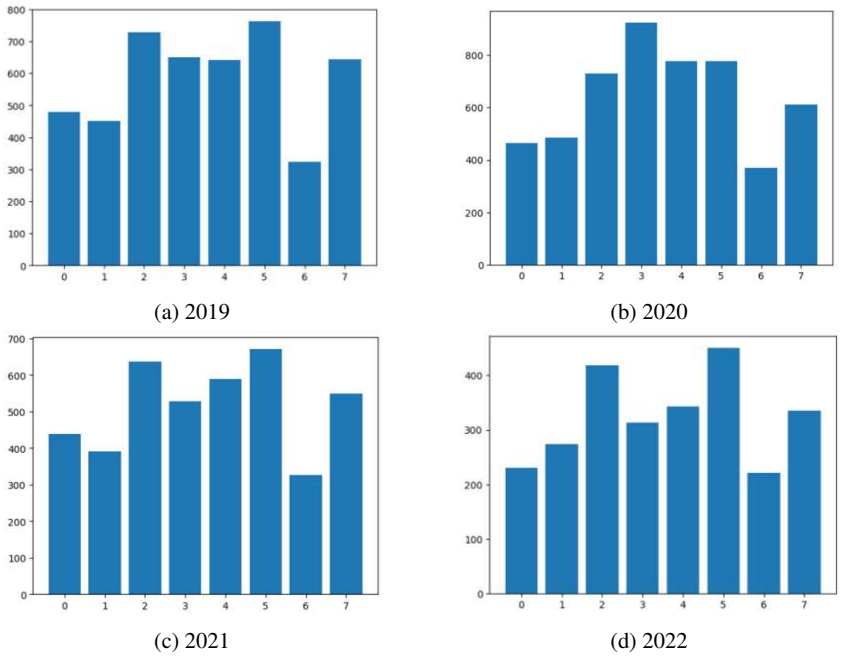
Table 4.2: Centrality, connectivity, path-based and mesoscopic measures of our dataset computed on different subgraphs. Specifically, G_AP and G_AI are bipartite graphs (with A and P, A and I resp., as node types). G_A is the inferred homogeneous directed unweighted graph, corresponding to the unique meta-path-based graph comprising all meta-paths. All other graphs are variant of *G_A*: *w* denotes the addition of edge weights, *all* denotes the isolated nodes not included in any meta-path instance, *attr* denotes the addition of numeric attributes for node type A, and *und* denotes the removal of edge direction.

Measure	**Value**	**Graph**
Average in-degree from P *	1.22	G_AP
Average in-degree from I *	0.26	G_AI
Average in-degree from A *	3.61	G_A
Average weighted in-degree from A *	4.17	G_A_w
Degree assortativity	0.97	G_A_w
Attribute assortativity (label)	0.39	G_A_attr
Attribute assortativity (n_works)	0.09	G_A_attr
Attribute assortativity (n_cit)	0.15	G_A_attr
Attribute assortativity (impact_factor)	0.18	G_A_attr
Attribute assortativity (h_index)	0.16	G_A_attr
Attribute assortativity (i10_index)	0.18	G_A_attr
Transitivity	0.96	G_A
Clustering coefficient	0.65	G_A
Density	0.000563	G_A
Average path length largest CC	13.01 (897 nodes)	G_A_und
Diameter largest CC	33 (897 nodes)	G_A_und
# Strongly connected components	2355	G_A
# Weakly connected components	2309	G_A
# Communities	2769 - 2327	G_A - G_A_und
Modularity	0.93 - 0.97	G_A - G_A_und

'book'), publication year >= 2019 and 'Computer science' as *concept* with score >= 0.5, from which we derived the connected entities. We used the year as our time

interval, and thus examine 4 time intervals, or rather 3 increments, from 2019 to 2022.

Each node type is associated with categorical, numeric, and/or textual attributes. Author attributes include full name, impact factor, h-index and i10-index, number of published papers, and number of received citations. Institution attributes include name, country, and type (e.g., education, company, or nonprofit). For papers, in addition to information used as a filter for scraping, the title, the locations (e.g., the journal or book title), the number of citations per year, the full abstract, and the weighted list of relevant concepts are provided. For attribute encoding, we employed one-hot encoding for categorical attributes, feature scaling for numeric attributes and SentenceBERT [71] (with all-MiniLM-L6-v2 as pretrained model) for text attributes; since the max sequence length is set to 256, we split long texts (such as abstracts) into chunks, encoded individual chunks and computed their mean. All node types have their attribute vectors of different sizes. Note that our dataset contains a mixture of static and dynamic attributes; for instance, the number of published works or the number of citations may vary each year.



(a) 2019 (b) 2020

(c) 2021 (d) 2022

Fig. 4.3: Distribution of classes for each of the 4 years. The label mapping is as follows: 0: 'Economics and Business', 1: 'Engineering', 2: 'Mathematics', 3: 'Natural/Earth Science and Medicine', 4: 'Philosophy and Art', 5: 'Physics', 6: 'Political science', 7: 'Psychology, Sociology and History'.

Table 4.3: Comparison of different models trained on the entire dataset spanning the years from 2019 to 2022 according to multiple evaluation metrics. The first column designates both the architecture and dataset employed. The best results are highlighted in bold, the second best are underlined.

Model	F1-micro	F1-macro	F1-weighted	ROC-AUC
MLP on only A-type features	0.145±0.000	0.036±0.000	0.042±0.000	0.515±0.000
GAT on homogeneous net with A-type nodes	0.339±0.001	0.332±0.001	0.337±0.001	0.701 ± 0.001
GAT on heterogeneous net with A-type nodes	0.560 ± 0.003	0.553±0.002	0.558±0.002	0.698±0.002
GAT on heterogeneous net with A-, P- and I-type nodes	**0.742±0.006**	**0.735±0.006**	**0.742±0.006**	**0.956±0.002**

We validated the proposed framework on a multi-class classification task, where the class attribute corresponds to the authors' first *concept* (i.e., main area of expertise), and the labels are 8 subfields of 'Computer science (CS)': 'CS.Economics and Business' (0),'CS.Engineering' (1), 'CS.Mathematics' (2), 'CS.Natural/Earth science and Medicine' (3), 'CS.Philosophy and Art' (4), 'CS.Physics' (5), 'CS.Political science' (6) and 'CS.Psychology, CS.Sociology and History' (7). The derivation of the classes exploited the OpenAlex 6-level hierarchy of concepts, which is a modified version of the tree structure proposed by [78], and the 'wikidata' attribute, which is the link to the associated Wikipedia page. The distribution of classes for each of the 4 years is shown in Figure 4.3.

4.3.2 Advantage of HINs

Network models, particularly heterogeneous networks, offer a holistic approach that exploits the interconnectedness of academic entities and their features, thus enabling more robust and informative predictions. With the aim of classifying authors' primary fields of study, we investigated different models by progressively enriching the feature space available for classification, thereby enhancing predictive accuracy. For each model, we conducted five independent runs for 500 epochs each on the entire dataset spanning the years 2019 to 2022, with training, validation and test sets comprising the same authors (70%, 15% and 15% split, respectively). Table 4.3 reports the mean and standard deviation values for micro F1-score, macro F1-score, weighted F1-score, and ROC-AUC.

Initially, we examined the results of classification based solely on author attributes, excluding any structural information derived from their relationships. We trained a simple MLP model with one hidden layer using the Adam optimization algorithm [38] with full batch size and learning rate set to 0.01. Our findings revealed that author attributes alone were inadequate for effective classification. We observed a marginal enhancement over a dummy model — representing completely random classification

aligned with class distribution) — particularly in the metrics accommodating class imbalance.

Subsequently, we incorporated structural information by modeling the data using a two-layer GAT over a homogeneous network, with Adam optimization algorithm and learning rate equals to 0.01. This network featured a single node type (Author) and a single relationship type, wherein a link exists between two authors if they are co-authors, affiliated with the same institution, or one cited the other. Table 4.3 shows a substantial increase in performance, justifying the adoption of a graph structure-based model.

Nevertheless, the proposed model still overlooks the various semantics inherent in relationships, each contributing distinctively to the classification task. Consequently, we explored a second network model — a first HIN model, following the definition $|A|+|R| > 2$ — maintaining authors as unique node type but distinguishing between the 3 relationship types. We kept the same optimization function and hyperparameters as the homogeneous model, while making the architecture explicitly discerning the contribution of different types of relationships. This enrichment led to a further performance improvement, although it neglected information embedded in other node types, specifically Papers and Institutions. The best results are achieved by a HIN model — an enriched HIN model, following the definition $|A| > 2$ & $|R| > 2$ which featured 3 node types, 3 edge types, and 3 meta-path types, as outlined in Section 4.3.1. The optimization algorithm and hyperparameters are the same as the previous model.

The semantic richness resulting from modeling multiple relationships and more specifically multiple node types, along with their specific information content, proved to be beneficial for the given task. As a consequence, we shall focus our study on graph continual learning on HINs using the latest and richest proposed model of heterogeneous GAT as an upper bound of our incremental model.

4.3.3 Competing methods

As we previously discussed (cf. Sections 2.4 and 4.1), direct competitors for DyHANE in GNN-based continual learning on HINs are missing or unable to accommodate dynamic attributes.

Having discussed the benefits offered by network models with multiple node and edge types, we compared our proposed incremental model with two baselines, the Retrained and Online models, on our multi-class classification task to assess the effectiveness and efficiency of our proposed approach. More specifically, we trained DyHANE at each timestamp on $\mathcal{T}^t = \mathcal{I}^t \bigcup \mathcal{B}^{t-1}$, i.e., on a training set comprising the detected nodes affected by changes and the nodes stored in the experience buffer, and compared it with the following methods:

- *Retrained*GNN, having the same architecture as DyHANE and $\mathcal{T}^t = \mathcal{V}^t$, i.e., trained on all nodes existing in the network at the given timestamp;

- *Online*GNN, having the same architecture as DyHANE and $\mathcal{T}^t = \mathcal{V}^t - \mathcal{V}^{t-1}$, i.e., trained on changed nodes only.

The two baselines employ the same GNN architecture as the proposed approach and reflect an upper bound in performance and execution time, respectively.

4.3.4 Experimental settings

We conducted our experiments on a Docker VM with 1x3090 GPU, 128GB of RAM, and 64 dedicated cores, on a 2x56-core Intel(R) Xeon(R) Gold 6258R CPU, with 256GB RAM and two NVIDIA GeForce RTX3090s, and OS Ubuntu Linux 22.04 LTS.

Although our model can handle different attribute vector sizes for different node types, we performed dimensionality reduction via Principal Component Analysis [65] on the attribute vectors and set all node dimensions to 128. As previously mentioned, we used a GATv2 [4] architecture as our GNN model for all experiments. Specifically, we implemented a two-layer GAT with hidden channels dimension set to 64 and out channel dimension set to 8 as the number of classes in our dataset (cf. Section 4.3.1). We employed a weighted cross entropy loss function, which addresses the class imbalance by assigning higher weights to underrepresented classes during the training procedure. We trained the model using the Adam optimization algorithm with full batch size, for 500 epochs for 5 independent runs, and we set the optimal hyperparameters for the learning process via grid search algorithm in the range of $\{0.05, 0.01, 0.005, 0.001\}$ for the learning rate, $\{0.005, 0.001, 0.0005, 0.0001\}$ for the weight decay and $\{0.2, 0.3, 0.4, 0.5, 0.6, 0.7\}$ for the dropout rate in combination with different configurations of the memory buffer. We carried out a grid search over $\{256, 512, 768, 1024, 2048\}$ for the buffer size $|\mathbf{B^t}|$ and tested two different buffer compositions, storing the experience nodes uniformly w.r.t their type or according to the importance of their type. Regarding the latter case, for the update of the memory buffer $\mathcal{B}^t$, we employed $top\text{-}k = 64$ nodes, for each type, to compute the type importance and select the number of nodes of that type to be stored as experience (cf Section 4.2.2). We found the best configuration as that corresponding to 0.01 as learning rate, 0.0001 as weight decay, 0.3 as dropout, and buffer size equal to 768 with different rates of nodes for each type summing to 1 (0.125 A, 0.77 P, 0.105 I).

4.3.5 Results

We organize the presentation of our experimental results into three parts. We first compare our best model (cf. Section 4.3.4) — hereinafter referred to as DyHANE$_{\mathcal{B}768}$ — with the two baselines in two different scenarios. We then delve into different configurations of our framework, focusing on different sizes and compositions of the

memory buffer and performing an ablation study of the memory buffer. Finally, we discuss the sensitivity analysis of the (hyper)parameters of our models.

4.3.5.1 Comparison with baselines

For a fair and meaningful comparison, we compare the results obtained by our best model and the baselines in two different scenarios, corresponding to two different test sets equals for all models:

- 'C': test set consisting only of changed nodes, corresponding to the Online model's test set;
- 'C+U': test set consisting of a set of both changed and unchanged nodes, selecting a subset of nodes that were neither identified by our algorithms as nodes affected by changes (Algorithm 2) nor as experience to be stored in the buffer (Algorithm 3).

In the first scenario, we assess the effectiveness of our method to learn new patterns; in the second scenario, we assess the capacity of our method to retain unchanged patterns while integrating new ones.

It should be noted that, in both scenarios, the changed nodes in the test set are new nodes of the last increment, i.e., they are 2022 nodes not existing in 2019/2021. The unchanged nodes are nodes existing before the last increment and not modified based on events of 2022. Furthermore, we examined the behavior of incremental models over multiple increments, i.e., 2019-2020, 2020-2021, and 2021-2022, to investigate performance degradation under sequential (continuous) application of the models. We show the results for different scenarios and number of increments under multiple evaluation metrics in Table 4.4, specifying the total execution time for each model. The total execution time corresponds solely to training time for baselines, while for our model it comprises both the detection of nodes affected by changes and training time. Time (in seconds) is the average over 5 independent runs of the model on a single increment; in the case of multiple increments, it is an average over all increments.

Comparing the overall performance of each model in the two different scenarios, we note that, as evidenced in Table 4.4, only DyHANE$_{\mathcal{B}768}$ improves performance on the 'C+U' scenario, i.e., when testing on a sample comprising both changed and unchanged nodes; the *Online* model fails to make correct predictions for unchanged nodes, while apparently the *Retrained* model suffers from the reduction of its training set.

To gain a comprehensive understanding of models' performances, we consider multiple evaluation metrics, including F1 measures and Area Under the Receiver Operating Characteristic Curve (ROC-AUC), which provides a summary measure of the model's ability to rank instances. A high ROC-AUC suggests good discrimination ability, while a lower F1-score is related to class imbalance; this applies especially to the macro-F1 score which treats all classes equally. Table 4.4, especially F1-macro exhibiting the lowest values for all models in each scenario, can be explained based

Table 4.4: Comparison with baselines using the same test set; scenario 'C' refers to the test set comprising only changed nodes, scenario 'C+U' refers to the test set comprising both changed and unchanged nodes. Incremental models are tested after 1 and 3 network increment(s). Time is expressed in seconds; for DyHANE$_{\mathcal{B}768}$, it consists of both the detection of nodes affected by changes and training time; for the baselines it corresponds solely to training time. The best results are highlighted in bold, the second best are underlined.

Scen.	#Incr.	Model	F1-micro	F1-macro	F1-weighted	ROC-AUC	Time
'C'	-	*Retrained*GNN	**0.742 ± 0.006**	**0.735 ± 0.006**	**0.742 ± 0.006**	**0.956 ± 0.002**	159.9
	1	*Online*GNN	0.431 ± 0.001	0.422 ± 0.001	0.429 ± 0.001	0.692 ± 0.001	**65.5**
	3	*Online*GNN	0.398 ± 0.001	0.386 ± 0.001	0.397 ± 0.001	0.651 ± 0.001	65.6
	1	DyHANE$_{\mathcal{B}768}$	0.664 ± 0.001	0.659 ± 0.001	0.664 ± 0.001	0.932 ± 0.001	78.9
	3	DyHANE$_{\mathcal{B}768}$	0.661 ± 0.001	0.655 ± 0.001	0.660 ± 0.002	0.918 ± 0.001	80.0
'C+U'	-	*Retrained*GNN	**0.712 ± 0.006**	**0.709 ± 0.006**	**0.712 ± 0.006**	**0.953 ± 0.003**	159.9
	1	*Online*GNN	0.285 ± 0.004	0.277 ± 0.003	0.281 ± 0.003	0.645 ± 0.002	**65.6**
	3	*Online*GNN	0.259 ± 0.004	0.252 ± 0.003	0.256 ± 0.003	0.592 ± 0.003	**65.6**
	1	DyHANE$_{\mathcal{B}768}$	0.676 ± 0.001	0.671 ± 0.001	0.676 ± 0.001	0.940 ± 0.001	78.9
	3	DyHANE$_{\mathcal{B}768}$	0.674 ± 0.001	0.669 ± 0.001	0.674 ± 0.001	0.937 ± 0.001	79.4

on the imbalance of our dataset, which features unbalanced increments and is not constrained to a constant number of changes over time.

Our framework always achieves worse performance w.r.t. the *Retrained* model, due to the smaller size of the training set and the higher class imbalance at each increment. The weakened difference between the ROC-AUC values, w.r.t the difference in F1-measures, confirms this hypothesis. Simultaneously, we observe to be more than twice faster in terms of execution time, considering that we include the calculation of the influenced node set in the elapsed time value.

On the contrary, DyHANE significantly outperform the Online GNN model in comparable time. We spot that it achieves the lowest performance on both scenarios, probably due to the smaller amount of data. Specifically, it dramatically underperforms when tested on unchanged nodes, demonstrating the inadequacy of only initializing the GNN parameters to retain the knowledge of the past network.

We specify that the generation of the influenced node set $\mathcal{I}^t$ at each new timestamp requires a maximum of 8.0 seconds. As previously pointed out (cf. Sections 4.2.1 and 4.2.3), the identification of nodes affected by change relies on the calculation of incremental meta-paths. Meta-path processing occurs in parallel, and the longest meta-path type in our dataset is A-P-P-A with length four, thus requiring the Cartesian product between two (sub)sets of edges. More specifically, if the event to be processed is an edge of type A-P, we intersect it with all of type P-P and then with all of type A-P; if the event to be processed is of type P-P, we intersect it twice with all of type A-P. We should also consider that authors typically work in groups and not alone, so we are unlikely to reach the worst case of checking all pairs of edges. The resuming time of the memory buffer $\mathcal{B}^{t-1}$ and the parameters θ^{t-1} learned at the previous timestamp is negligible. Moreover, it is noteworthy that the computation of $\mathcal{B}^{t-1}$ with the Explainer (cf. Sections 4.2.2 and 4.2.3) consistently concludes within a maximum time frame of 9.7 seconds.

We note that the execution time of our model in the case of multiple increments increases slightly. Recalling that the execution time reported in Table 4.4 is the average over the three increments and excludes the update of the memory buffer (which is computed while waiting for the next increment), the difference is due to processing several times, i.e., in multiple increments, some of the edges, and not just once as in the single-increment case. From the perspective of expressiveness, we notice only a slight degradation of the values in the case of multiple increments, which demonstrates the effectiveness of the proposed approach, especially in updating $\mathcal{B}^t$ from $\mathcal{B}^{t-1} \bigcup \mathcal{I}^t$, with $|\mathcal{B}^{t-1} \bigcup \mathcal{I}^t| \ll |G^t|$. *Online*GNN, conversely to our model, is significantly affected by multiple increments.

Finally, we observe remarkable stability of DyHANE$_{\mathcal{B}768}$ with respect to the baselines, as reflected by the low standard deviation in comparison with competitors.

Our proposed framework thus emerges as a promising trade-off between expressiveness and computational efficiency, positioning itself as a potential candidate for applications with limited or fixed computational budget. Achieving comparable performance w.r.t. our evaluation metrics on both scenarios, i.e., on both training sets, we can assess that using the Explainer to keep the most important nodes with the experience reply strategy is effective in retain previous knowledge and that the proposed strategy to detect changes leveraging the semantics of meta-paths helps in integrate knew knowledge.

4.3.5.2 Analysis of the DyHANE variants

We investigated multiple variants of our model, differing in the size of the memory buffer $|B^t|$, to gain deeper insights into its impact on our node classification task. More precisely, we conducted an ablation study by removing the memory buffer (equivalently, buffer size equals to 0), and then floated the size in the set $\{256, 512, 768, 1024, 2048\}$. The specific size for each model can be easily identified in the subscript appended to its name. For each buffer size, we experimented two different compositions: number of nodes for each type based on type importance (Cf. Algo 3) and same number of nodes for each type. In order to distinguish models with the same memory buffer size but different compositions, we marked with f the models using a uniform composition for different types, where f stands for *fixed-size of node types* as opposed to dynamic computation of type importance. To summarize:

- DyHANE$_{\mathcal{B}0}$, with $\mathcal{T}^t = \mathcal{I}^t$, i.e., trained only on the set of nodes directly or indirectly affected by changes
- DyHANE$_{\mathcal{B}256}$, with $\mathcal{T}^t = \mathcal{I}^t \bigcup \mathcal{B}^{t-1}$ and $|\mathcal{B}^{t-1}| = 256$, i.e, trained on the set of nodes affected by changes and experience nodes stored in a buffer of cardinality equals to 256 with floating composition w.r.t. different node types
- DyHANE$_{\mathcal{B}256f}$, with $\mathcal{T}^t = \mathcal{I}^t \bigcup \mathcal{B}^{t-1}$ and $|\mathcal{B}^{t-1}| = 256$ and $|A_{\mathcal{B}}| = |P_{\mathcal{B}}| = |I_{\mathcal{B}}|$, i.e, trained on the set of nodes affected by changes and experience nodes stored in a buffer of cardinality equals to 256 with uniform composition w.r.t. different node types

- DyHANE$_{\mathcal{B}512}$, with $\mathcal{T}^t = \mathcal{I}^t \bigcup \mathcal{B}^{t-1}$ and $|\mathcal{B}^{t-1}| = 512$, i.e, trained on the set of nodes affected by changes and experience nodes stored in a buffer of cardinality equals to 512 with floating composition w.r.t. different node types
- DyHANE$_{\mathcal{B}512f}$, with $\mathcal{T}^t = \mathcal{I}^t \bigcup \mathcal{B}^{t-1}$ and $|\mathcal{B}^{t-1}| = 512$ and $|A_{\mathcal{B}}| = |P_{\mathcal{B}}| = |I_{\mathcal{B}}|$, i.e, trained on the set of nodes affected by changes and experience nodes stored in a buffer of cardinality equals to 512 with uniform composition w.r.t. different node types
- **DyHANE$_{\mathcal{B}768}$**, with $\mathcal{T}^t = \mathcal{I}^t \bigcup \mathcal{B}^{t-1}$ and $|\mathcal{B}^{t-1}| = 768$, i.e, trained on the set of nodes affected by changes and experience nodes stored in a buffer of cardinality equals to 768 with floating composition w.r.t. different node types
- DyHANE$_{\mathcal{B}768f}$, with $\mathcal{T}^t = \mathcal{I}^t \bigcup \mathcal{B}^{t-1}$ and $|\mathcal{B}^{t-1}| = 768$ and $|A_{\mathcal{B}}| = |P_{\mathcal{B}}| = |I_{\mathcal{B}}|$, i.e, trained on the set of nodes affected by changes and experience nodes stored in a buffer of cardinality equals to 768 with uniform composition w.r.t. different node types
- DyHANE$_{\mathcal{B}1024}$, with $\mathcal{T}^t = \mathcal{I}^t \bigcup \mathcal{B}^{t-1}$ and $|\mathcal{B}^{t-1}| = 1024$, i.e, trained on the set of nodes affected by changes and experience nodes stored in a buffer of cardinality equals to 1024 with floating composition w.r.t. different node types
- DyHANE$_{\mathcal{B}1024f}$, with $\mathcal{T}^t = \mathcal{I}^t \bigcup \mathcal{B}^{t-1}$ and $|\mathcal{B}^{t-1}| = 1024$ and $|A_{\mathcal{B}}| = |P_{\mathcal{B}}| = |I_{\mathcal{B}}|$, i.e, trained on the set of nodes affected by changes and experience nodes stored in a buffer of cardinality equals to 1024 with uniform composition w.r.t. different node types
- DyHANE$_{\mathcal{B}2048}$, with $\mathcal{T}^t = \mathcal{I}^t \bigcup \mathcal{B}^{t-1}$ and $|\mathcal{B}^{t-1}| = 2048$, i.e, trained on the set of nodes affected by changes and experience nodes stored in a buffer of cardinality equals to 2048 with floating composition w.r.t. different node types
- DyHANE$_{\mathcal{B}2048f}$, with $\mathcal{T}^t = \mathcal{I}^t \bigcup \mathcal{B}^{t-1}$ and $|\mathcal{B}^{t-1}| = 2048$ and $|A_{\mathcal{B}}| = |P_{\mathcal{B}}| = |I_{\mathcal{B}}|$, i.e, trained on the set of nodes affected by changes and experience nodes stored in a buffer of cardinality equals to 2048 with uniform composition w.r.t. different node types

Table 4.5 provides for each model the total training set cardinality (which is the same for two models with equal memory buffer size) and detail for each node type, coupled with its execution time. The values refer to the last network increment (2022). We supplement the equivalent information for baselines to strengthen the comparison, observing that all our models lie between the two. The table emerges sorted in ascending order of training set size and execution time.

Table 4.5 shows that the node type with the most impact on classifying authors' primary field of study is P, although it is not the target node type. This is due to the higher information content of Paper attributes, including abstracts. Node type P is followed by A and I, respectively, in lower ratio.

To gain deeper insights into the impact of heterogeneous nodes experience replay on our node classification task, we conducted a comprehensive assessment involving all the aforementioned configurations. The evaluation of the multiple variants of DyHANE, each under its best configuration, is referred to Table 4.6.

Table 4.6 shows a surprising stability in all the results of our models, as indicated by the low standard deviation. A low standard deviation typically denotes that the performance of the classifier is consistent across different classes.

Table 4.5: Training set size and composition for baselines and variants of DyHANE, coupled with their mean execution time on the last network increment.The cardinalities of the training sets and running times are in ascending order. The training set discriminates between the different node types, (A)uthors, (P)apers and (I)nstitutions, with $|T| = |A| + |P| + |I|$. The training set of our models comprises both the nodes from $\mathcal{I}^{2022} \bigcup \mathcal{B}^{2021}$. Given the same buffer size, models marked with f have a different internal composition but same final training set size as their counterparts. The time is in seconds and refers to the best hyperparameters configuration of each model. For our models, it includes the detection of nodes affected by changes and the training time. Baselines at the extremes are highlighted in the boxes, our best model name is highlighted in bold.

Model	**|T|**	**|A|**	**|P|**	**|I|**	**Time (*sec*)**
Online GNN	4611	2584	1563	464	65.6
DyHANE$_{\mathcal{B}}$0	5751	3409	1817	525	73.0
DyHANE$_{\mathcal{B}256}$	6007	3441	2014	552	74.4
DyHANE$_{\mathcal{B}256f}$	6007	3495	1902	610	74.4
DyHANE$_{\mathcal{B}512}$	6263	3483	2194	586	75.7
DyHANE$_{\mathcal{B}512f}$	6263	3580	1987	695	75.7
DyHANE$_{\mathcal{B}768}$	6519	3505	2408	606	78.9
DyHANE$_{\mathcal{B}768f}$	6519	3665	2073	781	78.9
DyHANE$_{\mathcal{B}1024}$	6775	3537	2605	633	86.3
DyHANE$_{\mathcal{B}1024f}$	6775	3751	2158	666	86.3
DyHANE$_{\mathcal{B}2048}$	7799	3665	3393	741	95.2
DyHANE$_{\mathcal{B}2048f}$	7799	4022	2499	1208	95.2
Retrained GNN	27402	15397	10174	1831	159.6

We first performed an ablation study, removing the memory buffer from the training set, which thereby contains only the nodes affected by changes identified by Algorithm 2. We spotted that the model with buffer size equal to 0 records the lowest values among our models, but still considerably higher than the *Online* model. We can thus assess the effectiveness of the proposed approach in identifying the subset of nodes significantly related to changed nodes.

We then explored different sizes of the memory buffer. We observed that reducing the number of experience nodes leads to a degradation in the classification task while speeding up the training process, prompting to consider a trade-off. We spotted a satisfactory balance in correspondence to size 768. By increasing the buffer cardinality to 1024, the difference in scores is not adequate to justify the observed delay (we recall that a maximum of 8.0 seconds is required for the identification of nodes affected by changes); by reducing the buffer cardinality to 512, the saved time is not adequate to justify remarkable performance degradation for all evaluation metrics. We emphasize that although there are few nodes in the buffer, their identification based on Algorithm 3 has proven to be effective in selecting meaningful samples.

We noticed that the Explainer is successful also in calculating the importance of the contribution of different node types. More specifically, the comparison of pairs of models with the same buffer cardinality but different composition, i.e., models

Table 4.6: Comparison of multiple DyHANE models in the 'C+U' scenario, with baselines at the extremities highlighted in the boxes and our best model name highlighted in bold. A model is identified by the number shown as subscript in its name which refers to the size of the memory buffer, and the suffix f refers to a possible fixed composition of the buffer, with equal number of nodes for each type. Time is expressed in seconds and comprises both the detection of nodes affected by changes and training time.

Model	F1-micro	F1-macro	F1-weighted	ROC-AUC	Time
*Online*GNN	0.285 ± 0.004	0.277 ± 0.003	0.281 ± 0.003	0.645 ± 0.002	65.6
DyHANE$_{\mathcal{B}0}$	0.561 ± 0.002	0.554 ± 0.001	0.559 ± 0.002	0.817 ± 0.002	73.0
DyHANE$_{\mathcal{B}256f}$	0.652 ± 0.002	0.644 ± 0.001	0.652 ± 0.001	0.880 ± 0.001	74.4
DyHANE$_{\mathcal{B}256}$	0.663 ± 0.002	0.656 ± 0.002	0.662 ± 0.002	0.891 ± 0.001	74.4
DyHANE$_{\mathcal{B}512f}$	0.665 ± 0.001	0.659 ± 0.001	0.666 ± 0.001	0.896 ± 0.001	75.7
DyHANE$_{\mathcal{B}512}$	0.671 ± 0.002	0.667 ± 0.001	0.671 ± 0.001	0.903 ± 0.001	75.7
DyHANE$_{\mathcal{B}768f}$	0.672 ± 0.002	0.668 ± 0.001	0.672 ± 0.002	0.919 ± 0.001	78.9
DyHANE$_{\mathcal{B}768}$	0.676 ± 0.001	0.671 ± 0.001	0.676 ± 0.001	0.940 ± 0.001	78.9
DyHANE$_{\mathcal{B}1024f}$	0.676 ± 0.001	0.671 ± 0.001	0.677 ± 0.001	0.941 ± 0.002	86.3
DyHANE$_{\mathcal{B}1024}$	0.678 ± 0.001	0.672 ± 0.002	0.678 ± 0.002	0.942 ± 0.001	86.3
DyHANE$_{\mathcal{B}2048f}$	0.680 ± 0.001	0.674 ± 0.001	0.681 ± 0.001	0.944 ± 0.002	95.2
DyHANE$_{\mathcal{B}2048}$	0.683 ± 0.002	0.677 ± 0.002	0.683 ± 0.002	0.949 ± 0.001	95.2
*Retrained*GNN	0.712 ± 0.006	0.709 ± 0.006	0.712 ± 0.006	0.956 ± 0.002	159.9

with and without f, shows that prioritizing one node type over another results in improved performance.

4.3.5.3 Sensitivity analysis

We performed sensitivity analysis of our main (hyper)parameters to ensure the robustness and reliability of the models, assessing the impact of variations in input parameters on model outputs and providing insights into the relative importance of different variables.

We first explored 100 DyHANE configurations varying the size of the memory buffer (for short, B_size) and two significant GAT's hyperparameters: dropout and learning rate, with weight decay (also known as L2 regularization) set to 0.0001 in all the experiments (cf. Section 4.3.4). The F1-weighted score for each triplet is shown in the scatter plot in Figure 4.4.

The combination of the highest learning rate and the highest dropout values (0.05 and 0.7, resp.) resulted in extremely poor performances, thus we preferred to remove them from the plot in order to improve readability. More generally, higher dropouts (0.7, 0.6 and even 0.5) and learning rates at the extremes (the highest value 0.05 and the lowest 0.001) lead to poor performance. For each buffer size, the highest scores correspond to lower values of dropout (0.3 or 0.4) and learning rate in the range $[0.005, 0.01]$, hinting at the opportunity to limit regularization and balance convergence speed and stability during training. We recall that our training set at

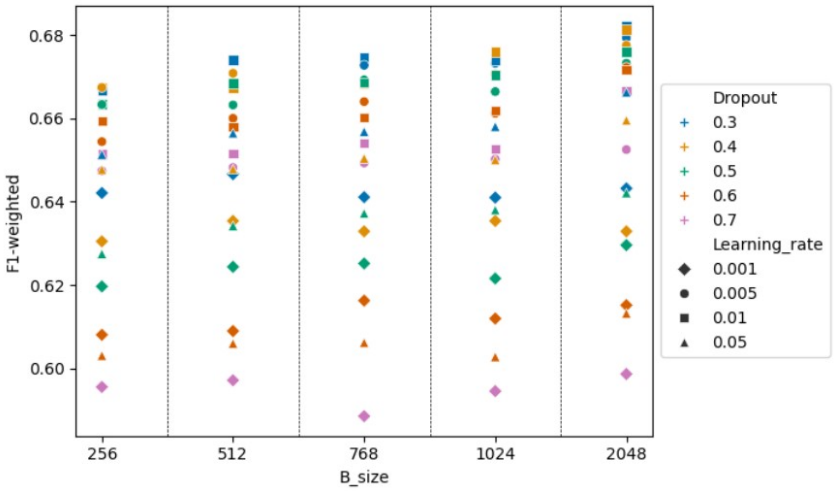


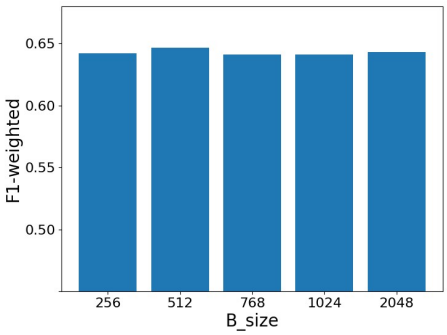
Fig. 4.4: Scatter plot of 100 DyHANE configurations. Floating the size of the memory buffer, we show the F1-weighted score based on dropout and learning rates. Different colors and symbols denote different dropout values and learning rates, respectively. The weight decay is set to 0.0001 in all the experiments. For visualization reasons, we removed from the plot the values with dropout 0.7 and learning rate 0.05, whose performance is below the plot limit.

each increment is a subset of the entire network and may exhibit new patterns even completely different from previous ones. We therefore need low regularization values (both dropout and weight decay) for learning new patterns, but very low values result in the risk of overfitting, reason why we do not explore values below 0.3 for dropout and 0.0001 for weight decay.

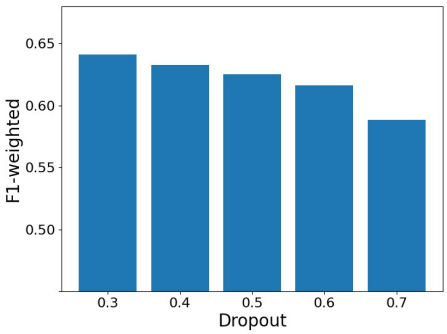
We elaborate further on the behavior of single hyperparameters. Our analysis was performed by varying the value of a single parameter at time while maintaining the others to our best configuration (cf. Section 4.3.4). This involved our main (hyper)parameters, i.e., the size of the memory buffer and three significant GAT hyperparameters: dropout, learning rate and weight decay. Figure 4.5 shows the F1-weighted score for each combination.

When varying the number of experience nodes (Figure 4.5(a)), we observe small fluctuations on F1-weighted values. Indeed, low values of weight decay (specifically, 0.0001) and dropout (0.3 and 0.4), and learning rates in the range $[0.005, 0.01]$ exhibited the best performance for all our models.

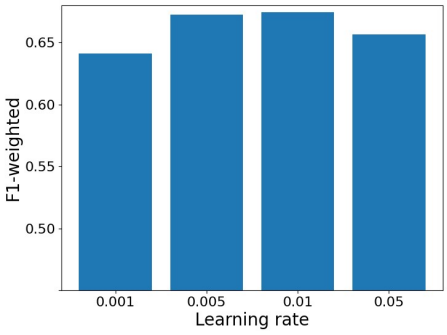
By varying the dropout (Figure 4.5(b)), we note that performance degrades as the fraction of node or edge features set to zero increases during each training iteration, i.e., as the dropout increases. This is not always true in GAT-based models, so that the *Retrained* model uses a dropout of 0.6. This might be due to the reduced size of the training set, since higher dropout values with fewer examples may introduce



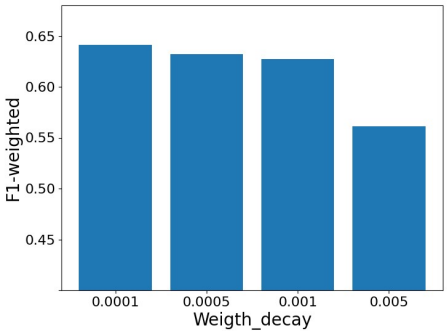
(a) Learning rate: 0.01, dropout: 0.3, weight decay: 0.0001.



(b) B_size: 768, learning rate: 0.01, weight decay: 0.0001.



(c) B_size: 768, dropout: 0.3, weigth decay: 0.0001.



(d) B_size: 768, dropout: 0.3, learning rate: 0.01.

Fig. 4.5: Sensitivity analysis of our main (hyper)parameters w.r.t. DyHANE best configuration (cf. Section 4.3.4). The x-axis indicates the investigated parameter, while the y-axis indicates the corresponding F1-weighted value. The other parameters are fixed for each barchart.

excessive regularization, hindering the model's ability to effectively capture the complex relational structures underlying the network.

Concerning the learning rate (Figure 4.5(c)), we identified the optimal values in the range $[0.005, 0.01]$, striking a balance between convergence speed and stability during training.

As regards the weight decay (Figure 4.5(d)), we note that performance degrades as large weights are more heavily penalized during training, i.e., as the weight decay increases. We found the optimal value at 0.0001, a relative low value which enables the model to learn complex patterns from the data without being overly constrained by regularization. Further decreasing this value may increase the risk of overfitting, since we are dealing with a reduced dataset corresponding to an increment.

Remarks on regularization strategies. Note that the number of influenced nodes, varying at each timestamp, is usually significantly larger than the size of the experience buffer, which is instead of fixed size. To cope with the overfitting problem caused by the small number of replayed nodes, a commonly used strategy is to add an extra regularization term to the loss function to guarantee that the distance between the current and the historical model parameters will not deviate further; more specifically, different importance is given to different parameters to keep small the changes of GNN parameters that are important to the past network while the others can be updated more drastically. We found that the Elastic Weight Consolidation (EWC) regularization-based method [41] improve performance in all models so, for the sake of model comparison, we avoid to integrate any additional regularization term in our current formulation.

4.4 Concluding remarks

Incremental GNNs and HINs represent a totally unexplored field of research. In this regard, we present DyHANE, a GNN-based incremental framework capable of handling multiple types of nodes and relationships in a dynamic scenario. DyHANE adapts to changes in network topology and node attributes efficiently, by updating GNN parameters and training on a sample of the network. Specifically, we propose two algorithms, one for identifying the (minimum) set of nodes affected by changes and the other for identifying the (minimum) set of nodes to be used as experience node replay. On a multi-class classification task, we demonstrate the advantages of modeling offered by HINs and show the ability of our framework to achieve good performance on both changed and unchanged nodes, in a comparable time to online GNNs that suffer of performance degradation on unchanged nodes.

In the future, we are interested in investigating and comparing the results obtained in various dynamic scenarios, including a stable distribution of classes, skew, abrupt shift, and concept drift. We intend to test DyHANE on new datasets, such as predicting the primary interest(s) of users in a social network, and on new tasks, like predicting the number of papers'/authors' citations, or co-authorship relationships. The flexibility of our pipeline will also enable further optimisation of the proposed algorithms for detecting nodes affected by changes and experience nodes on heterogeneous graphs of growing size.

Chapter 5
Graph mining for the evolution of social debate on climate crisis

Social media have widely been recognized as a valuable proxy for investigating users' opinions by echoing virtual venues where individuals engage in daily discussions on a wide range of topics. Among them, climate change is gaining momentum due to its large-scale impact, tangible consequences for society, and enduring nature. In this work, we investigate the social debate surrounding climate emergency, with particular emphasis on the Conference of the Parties (COP), the foremost global forum for multilateral discussion on climate-related matters. To this aim, we leverage graph mining and text mining techniques to analyze a large corpus of tweets spanning 7 years, aiming to uncover the fundamental patterns that underlie the climate debate. Our contribution in this work is manifold: (i) we provide insights into the key social actors involved in the climate debate and their relationships, (ii) we unveil the main topics discussed during COPs within the social landscape, (iii) we assess the evolution of users' sentiment and emotions across time. The proposed approach [54] is presented from the perspective of disaster management, to provide valuable support for strategic and operational decision-making and exhibits the potential to scale up to several emergency issues, highlighting its versatility and potential for broader use in analyzing and understanding the increasingly debated emergent phenomena.

5.1 Introduction

In recent years, the emergence of social media platforms inexorably changed the way we communicate, share information, and engage with large-scale events. Among these platforms, Twitter stands out as one of the most adopted to date and has been widely exploited as a source of information for researchers seeking to study real-time events and gather insights from them. Both public figures and ordinary individuals create and share content, resulting in a democratization of information that embraces a wide range of perspectives. Specifically, the analysis of Twitter data has been instrumental in understanding and responding to extreme events, ranging

from natural disasters to social crises. Such an approach has led to the emergence of a phenomenon commonly dubbed "people as sensors," i.e., to leverage the vast amount of user-generated social content to gain insights into the occurrence, impacts, and response to events.

Climate change represents one of the most enduring debates on the Twitter landscape, making it fertile land for researchers from different areas. In this regard, Twitter has been proven particularly influential in the context of the annual Conference of the Parties, i.e., the meeting related to the implementation of the United Nations Framework Convention on Climate Change (UNFCCC). Such meetings generate unprecedented debates involving a wide plethora of participants, from decision-makers to activists, thus making Twitter an essential platform for expressing opinions, raising awareness, organizing campaigns, and promoting initiatives, fostering the dynamic exchange of opinions on climate-centered topics. As a result, such a platform has become a promising tool for monitoring public opinion and analyzing the main trends, sentiments, and reactions that accompany the social debate around COPs.

By recognizing Twitter as a valuable information source, our goal is to study the evolution of the social debate around the climate crisis, in correspondence with the annual COP meetings, by leveraging social traces left by people discussing climate-related issues as a proxy for real-world debate. In this chapter, we exploit a combination of *Graph Mining* and *Text Mining* techniques seeking to unveil the main patterns and key actors that drive the debate, as well as delve into the sentiment and emotions that distinguish the social response to the climate change narrative on Twitter.

Plan of the chapter. The remainder of this chapter is structured as follows. Section 5.2 discusses related studies, Section 5.3 introduces the data employed in our analysis, Section 5.4 presents methodology and techniques, and Section 5.5 describes related experimental results; finally Section 5.6 contains concluding remarks and pointers for future research.

5.2 Related studies

Social media have provided researchers with an unprecedented opportunity to study global-scale phenomena and extreme events. Indeed, the distance between social and real debates is getting smaller and smaller, and people interacting on social networks are widely recognized as helpful *social sensors* acting as valuable sources of information when it comes to extreme events [74, 72, 59]. Climate change makes no exception, proving to be one of the most studied topics in the social sphere. In this regard, Kirilenko et al. [40] used Twitter data to investigate the linkage between people's sensory experiences of local temperature and climate change, also assessing the potential influence of mass media on this process. Geo-tagged Twitter data was studied through topic modeling and sentiment analysis by Dahal et al. [12] to characterize the climate change discussion between different countries

and over time. Cody et al. [10] assessed how collective sentiment varies in response to climate change news and events, unveiling the role of Twitter as a medium for spreading climate change awareness. By analyzing the tweets about the 2013 IPCC report, Pearce et al. [64] characterized the emerging communities of users around the debate, hinting that contrasting views might lead to greater interactions. In this regard, polarization around the climate social debate has been widely investigated in recent years [30, 15]. Tyagi et al. [85] studied 100 weeks of Twitter discourse about climate change to unveil that deniers of climate change tend to be more hostile towards people who believe in it, and vice versa. Falkenberg et al. [18] studied the Twitter discussion around the United Nations Conference of the Parties on Climate Change (COP), shedding light on increasing ideological polarization, driven by right-wing activity. Network analysis approaches have also been employed to study social media users discussing climate change. For instance, Williams et al. [97] exploited Twitter data to reveal that users tend to segregate within like-minded communities, i.e., echo chambers. Finally, Effrosynidis et al. [17] exploited 15M tweets to provide a comprehensive overview of climate change through several investigation aspects.

Compared to the above works, ours focuses on the development of the social debate regarding the climate crisis related to the annual COPs, through graph mining and text mining tools. Unlike [18] which is the closest study to us, we do not limit to understand polarization as we want to analyze the evolution of the social debate over the years by investigating the main dynamics around user engagement, the key topics discussed by them, and how sentiment and emotions evolve as an effect of the major events that characterize each COP.

5.3 Data

In order to study a phenomenon of such magnitude as climate emergence, it is necessary to have large amounts of data, which must be representative and ductile so as to allow for the proper extraction of knowledge. In this regard, we leveraged the most representative dataset involving climate social debate so far [18]. It consists of a large corpus of tweets spanning 7 years, from COP20 (2014) to COP26 (2021), gathered through the corresponding set of hashtags "cop2x", with $x \in \{0, ..., 6\}$, by considering a time windows of six months before and after the conference date. We hydrated the corresponding Tweet IDs through the Hydrator tool,[1] following the official Twitter guidelines. According to the recent changes to the latter, we point out that the most recent COP27 and COP28 are not part of the dataset, and it was not possible to collect the corresponding data at the time of writing this work due to the new policies enforced for Twitter Academic APIs. [2]

Once obtained all raw data, we processed it as follows. First, we filtered out all non-English tweets to ensure high applicability through major NLP tools to date.

[1] https://github.com/docnow/hydrator

[2] https://www.theverge.com/2023/5/31/23739084/twitter-elon-musk-api-policy-chilling-academic-research

Table 5.1: Dataset statistics: tweets from 1 week before to 1 weak after the conference dates. Each date is in the format $mm/dd/yyyy$.

	Conference dates	# Tweets	# Users
COP20	12/01/2014 - 12/12/2014	188 085	50 708
COP21	11/30/2015 - 12/12/2015	1 610 105	366 176
COP22	11/07/2016 - 11/18/2016	325 191	90 251
COP23	11/06/2017 - 11/18/2017	357 612	86 899
COP24	12/02/2018 - 12/14/2018	269 906	87 650
COP25	12/02/2019 - 12/13/2019	324 710	118 774
COP26	10/31/2021 - 11/12/2021	2 266 117	644 752

Then, to reduce potential noise in the COP social debate, we narrowed our focus to a smaller time window, keeping only tweets published between one week before and after the conference period. For text mining tasks, we pre-processed the tweet corpora, including cleaning and normalization operations, such as turning emojis into the corresponding text. To deepen sentiment and emotions of users, we split each COP dataset into three distinct groups representing the periods before, during, and after the conference. For graph mining tasks, we gathered all the data about *retweets*, filtering out users who have never retweeted posts from other users or have never been retweeted by other users, and all the data about *replies*, filtering out users who have never replied to other users or have never been replied by other users. The network that results from the union of all direct interactions, i.e., retweet and replies, between pairs of users is hereinafter referred to as **interaction network**.

We summarize the main details on individual COPs in Table 5.1. Overall, we spotted over 1.5M of tweets involving ∼1.2M unique users across all COPs, with a small subset of ∼3K users appearing in all COPs.

We also carried out a number of steps for the categorization of user profiles. We first identified profiles corresponding to international *organizations*, being them politically recognized or not, active in specific fields or generally contributing to the climate debate. We differentiated them from *news spreaders*, which include both reputable news agencies and thematic blogs, and the *official conference accounts* associated with each COP. We also differently labeled individual user accounts, mostly public figures to distinguish between *representatives of* well-known national and international *organizations*, *activists* of any age and gender, *volunteers*, *academics*, *politicians*, *journalists*, *artists* and *actors*. Finally, we highlighted *super-users* as those highly active in the social debate but not holding authoritative positions.

5.4 Proposed approach

Given an input corpus of tweets discussing COPs, we exploit a combination of graph mining and text mining approaches to discover valuable information that might help both decision-makers and users improve awareness of and better deal

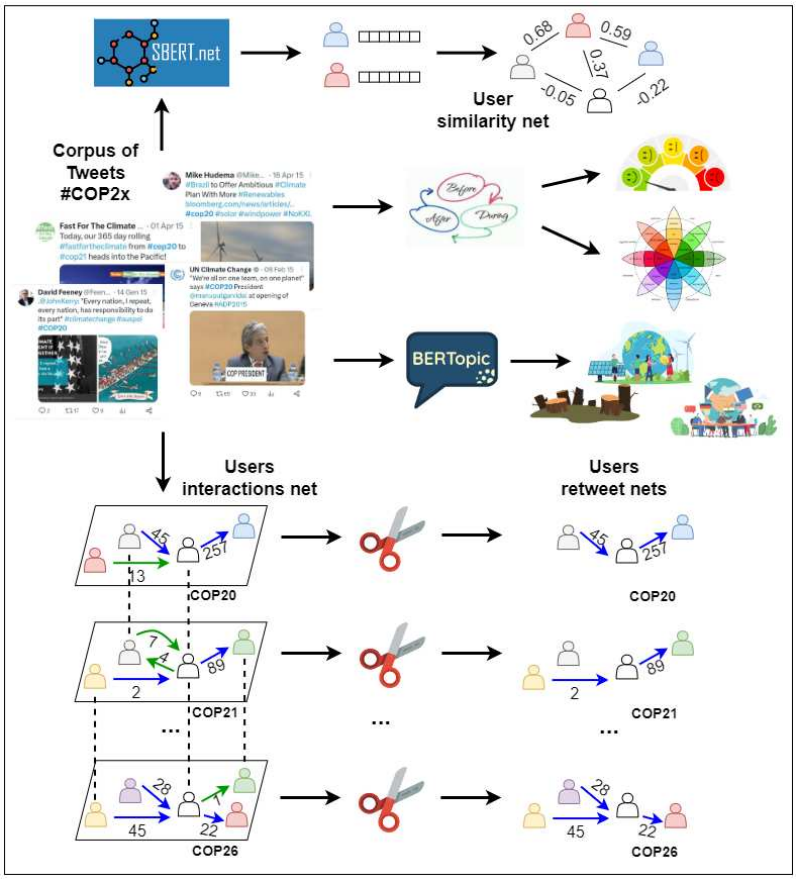


Fig. 5.1: Overview of the proposed approach, combing graph mining and text mining techniques.

with extreme events such as the climate crisis. Specifically, (i) we infer graph-based models to characterize the main framework underlying the social interactions and shed light on the most influential users for each COP; (ii) we leverage topic modeling to extract the most discussed topics for each COP; (iii) we use lexicon-based frameworks for shaping polarity and emotions expressed by users via textual components, to investigate how they evolve over time. An overview of the proposed modular approach is depicted in Figure 5.1.

The proposed modular approach supports a systematic, specialized, and comprehensive analysis of complex social data. The separation of the analysis into distinct modules enables specialized focus within each domain, fostering more effective and efficient analysis tailored to specific tasks, and enhances flexibility and scalabil-

ity, facilitating easier modification or updating of individual components without disrupting the entire system.

The three main modules for network analysis, topic modeling, and affective computing are described next.

5.4.1 Network analysis module

We exploit the information concerning retweets made available by Twitter to infer graph-based models to analyze main social interaction dynamics around COPs.

Network modeling. We are given, for each COP $i \in [20..26]$, a corpus $\mathcal{T}_i = \{t_{i,1}, ..., t_{i,n}\}$ containing all tweets written during the i-th COP, including one week before and after the conference dates.

For each COP i, we infer a directed-weighted heterogeneous *interaction* network $G_i = \langle \mathcal{V}_i, \mathcal{E}_i, R, \varphi, w \rangle$ such that $\mathcal{V}_i$ contains all users who *posted* or *retweeted* or *replied to* tweets $\in \mathcal{T}_i$ and $\mathcal{E}_i = \{(u, v) \mid u, v \in \mathcal{V}_i \wedge (u \text{ retweeted } v \vee u \text{ replied to } v)\}$; $R = \{ret, rep\}$ $\forall i$ is the set of relation types between user node pairs, $\varphi : \mathcal{E}_i \rightarrow R$ is the edge-type mapping function, and $w : \mathcal{E}_i \rightarrow \mathbb{N}^+$ is the weighing function that assigns each edge $(u, v) \in \mathcal{E}_i$ with a value corresponding to the number of retweets, resp. replies, made by user u on tweets posted by user v during the i-th COP period. Distinguishing between the two relation types, we obtain, for each COP i:

- a directed-weighted *retweet* network $G_i^{ret} = \langle \mathcal{V}_i^{ret}, \mathcal{E}_i^{ret}, w \rangle$ such that $\mathcal{V}_i^{ret}$ contains all users who *posted* or *retweeted* tweets $\in \mathcal{T}_i$ and $\mathcal{E}_i^{ret} = \{(u, v) \mid u, v \in \mathcal{V}_i \wedge \varphi(u, v) = ret\}$, i.e., *u retweeted v*;
- a directed-weighted *reply* network $G_i^{rep} = \langle \mathcal{V}_i^{rep}, \mathcal{E}_i^{rep}, w \rangle$ such that $\mathcal{V}_i^{rep}$ contains all users who *posted* or *replied to* tweets $\in \mathcal{T}_i$ and $\mathcal{E}_i^{rep} = \{(u, v) \mid u, v \in \mathcal{V}_i \wedge \varphi(u, v) = rep\}$, i.e., *u replied to v*.

The weighing function $w : \mathcal{E}_i^{ret} \rightarrow \mathbb{N}^+$, resp. $w : \mathcal{E}_i^{rep} \rightarrow \mathbb{N}^+$, assigns each edge $(u, v) \in \mathcal{E}_i^{ret}$, resp. $\mathcal{E}_i^{rep}$, with a value corresponding to the number of retweets, resp. replies, made by user u on tweets posted by user v during the i-th conference period. The set of nodes and edges in the interaction network comprise both retweets and replies, i.e., $\mathcal{V}_i = \mathcal{V}_i^{ret} \bigcup \mathcal{V}_i^{rep}$ and $\mathcal{E}_i = \mathcal{E}_i^{ret} \bigcup \mathcal{E}_i^{rep}$.

Structural network measures. Retweet networks are valuable sources of information for studying social phenomena such as engagement patterns, the spread of information, identifying influential users, and investigating the dynamics of online conversations [20]. In this respect, we exploit a wide range of structural and centrality network measures, as defined in Section 2.1.

Centrality measures provide multiple perspectives on the importance, influence, and connectivity of nodes (i.e., users) within a network, allowing shaping the roles of individuals w.r.t. the entire network. The average degree of a network provides an indication of the overall connectivity in the network by counting the average number of edges connected to each node; it can be declined into in-degrees, resp. out-

degrees, to consider only the number of incoming, resp., outgoing links. The former act as a proxy for nodes' popularity or prominence, while the latter represents the extent to which nodes spread information or influence peers in the network. Degree assortativity assesses the tendency of nodes to connect with similar nodes based on their degree (i.e., relevance properties). Sources (i.e., nodes without incoming edges) and sinks (i.e., nodes lacking outgoing edges) help to analyze the information flow in the network, by identifying the origin and accumulation points.

Connectivity measures are essential to describe link formation dynamics. Specifically, reciprocity, or dyadic closure, measures the extent to which connections in a network are reciprocated, indicating bilateral relationships. Triadic closure quantifies the tendency of nodes to form clusters or groups; in particular, transitivity assesses clustering tendency across the entire network by quantifying the proportion of connected triads (sets of three nodes) that also form closed triangles, whereas (local) clustering coefficient measures the clustering tendency of individual nodes by determining how strongly connected are the neighbors of a node. Finally, density quantifies the proportion of existing connections in a network relative to the total possible connections.

Path-based measures also describe efficiency and reachability aspects in a network. The average path length measures the average number of edges required to travel between any two nodes in a network. The diameter of a network is the maximum length of the shortest paths between any pair of nodes in the network, i.e., the maximum number of edges that must be traversed to go from one node to another.

Mesoscopic measures characterize the network cohesion in terms of substructures underlying the network. Particularly, strongly connected components are subgraphs where there is a directed path between any pair of nodes, highlighting cohesive groups, whereas in weakly connected components edge orientation is disregarded. Communities indicate the existence of particular subsets of nodes having denser internal connectivity compared to the external one. Modularity evaluates the quality of the partitioning into communities (or modules) by quantifying the degree to which nodes within the same module have more connections with each other compared to nodes in other modules.

5.4.2 Topic modeling module

We leverage text analysis of tweets to infer the trending topics and most prominent discussions in the social debate around COPs. Moreover, through this stage of topic modeling, clusters of topically-related tweets can be uncovered. In this respect, we resort to *BERTopic* [23], a powerful method employing BERT (Bidirectional Encoder Representations from Transformers) embeddings and c-TF-IDF (class-based Term Frequency-Inverse Document Frequency) to create dense clusters sharing the same topic. More precisely, BERTopic extracts deep semantic features from the input texts (i.e., tweets) using BERT as encoder, then the BERT-generated embeddings are clustered into similar groups based on the HDBSCAN clustering algorithm. From

each of the clusters, topics are finally represented as bags-of-words based on a cluster-level variant of the TF-IDF term relevance weighing function, able to incorporate class information into the traditional algorithm by capturing the discriminative power of terms specific to each class.

BERTopic succeeds in uncovering meaningful and coherent patterns and structures in textual data without any prior knowledge or explicit labeling, and allows for easily interpretable topics whilst keeping important words in the topic descriptions.

5.4.3 Affective computing module

Affective computing concerns the analysis of human affects, such as feelings and emotions, based on the computational treatment of subjectivity in a text. The proposed module is designed to detect and measure the sentiments and emotional states expressed by the users in their posted tweets, as detailed below.

5.4.3.1 Sentiment analysis

We carried out sentiment analysis through the *Valence Aware Dictionary and sEntiment Reasoner* (VADER) tool [28]. VADER is a versatile, rule-based lexicon sentiment classification method, specifically designed to measure the polarity in a social media text, i.e., to determine if the text expresses a positive, negative or neutral opinion. To this purpose, VADER maps lexical features to emotion intensities, i.e., sentiment scores, being able to understand the basic context of cue words and to understand the emphasis of capitalization and punctuation. The sentiment score of an input text can eventually be obtained by summing up the intensity of each word in the text.

We specify that VADER is a non-contextualized tool, thus it focuses solely on the intrinsic value of a text element while disregarding its contextual value with respect to the topic being discussed.

5.4.3.2 Emotion recognition

We also resort to *EmoAtlas*[3] for extracting, analyzing, and visualizing emotional information that characterizes an input text expressing the social debate related to COPs. EmoAtlas combines psychological lexicons, network modeling, and artificial intelligence to shape syntactic relationships between words in a text in 18 different languages and detect 8 different categorical emotions. Such emotions are conceived as follows: *joy* is a feeling of happiness, and the opposite of *sadness*, which is a state of sorrow or unhappiness; *fear* is triggered by the perception of danger or threat,

[3] https://github.com/alfonsosemeraro/emoatlas

and is the opposite of *anger*, the feeling of displeasure or rage; *anticipation* is the act of looking forward to or expecting something, and is the opposite of *surprise*, a feeling of astonishment or disbelief; *disgust* is a feeling of aversion or revulsion, and is the opposite of *trust*, a sense of faith, confidence, or reliance. A complex emotion can be conceived as a combination of multiple basic emotions, which are considered atomic and irreducible.

By means of EmoAtlas, input texts are first processed and enriched, then structured as a semantic network [81], and a score is assigned to each extracted emotion, in order to determine the prevailing sentiment and its magnitude compared to others. The visualization of emotions is carried out using the wheel layout, following the principles of proximity and opposition between pairs of emotions [69]. Emotions are then displayed such that their spatial adjacency or opposition in the wheel respectively indicates semantic proximity and semantic opposition expressed in the text.

5.5 Experimental evaluation

In this section, we describe the experimental results of our analysis. Through rigorous examination of each module's outcomes, this section elucidates the multifaceted nature of online community dynamics, unveiling intricate patterns of interaction, prevalent discussion themes, and evolving emotional expressions. By synthesizing findings from these distinct analytical modules, a comprehensive understanding emerges, shedding light on the nuanced interplay between social structures, thematic content, and user sentiment surrounding the social debate.

5.5.1 Interaction network analysis

We started our analysis by first considering an opportunity of modeling the interaction network from a multilayer perspective. Unfortunately, we found out a poor overlap of users between adjacent COPs, which averages around 12-13% for retweets, and 5-7% for replies. This is actually not surprising, since interactions in social networks are known to be sparse [62], as Table 5.2 shows for each individual layer. These findings prompted us to focus on individual COPs.

Table 5.2 also highlights the imbalance between the two types of relationship. We notice a lower number of replies compared to retweets. Removing the restriction on dates, i.e., retaining all tweets in the year about the conferences, we spot that the volume of replies during and in proximity of conference days is particularly low compared to all tweets about that same COP, standing between the maximum at COP20 (35%) and the minimum at COP25 (4%). It is likely that, during the conferences and in the days immediately preceding and following, attendees are engaging in real-time discussions and interactions through event-specific platforms, apps, or live streams rather than on broader social networks like Twitter. Furthermore, be-

Table 5.2: Main structural traits of the networks inferred from the COP data, where int_net refers to the interaction network, ret_net is the retweet network, and ret_rep the reply network.

	COP20	COP21	COP22	COP23	COP24	COP25	COP26
# Nodes int_net	44 839	325 397	82 089	78 025	74 636	106 450	567 952
# Edges int_net	97 564	820 213	163 173	160 061	138 370	195 485	1 361 186
# Nodes ret_net	44 217	324 186	81 864	77 528	73 951	106 274	565 500
# Edges ret_net	95 350	818 433	162 447	158 624	136 895	195 070	1 354 656
# Nodes rep_net	2 308	2 466	1 016	1 783	1 861	673	6 825
# Edges rep_net	2 801	3 011	867	1 729	1 667	472	6 871

fore and after the conferences, there may be anticipation and follow-up discussions, such as promotional activities or discussions building up excitement, and recaps, reflections and discussions about the outcomes, respectively, which could lead to higher engagement. The retweet rate, on the other hand, is never below 50% of the whole set of tweets about the same COP, with peaks at COP21, COP22 and COP26 above 64%. During the days of the conferences and the surrounding weeks, there may be a heightened buzz and excitement surrounding the event. The reasons can be manifold: an increased interest of people to share immediate event-related content with their followers, amplifying the event's reach, rather than be engaged in protracted online discussions; dissemination of valuable or timely information, such as announcements, updates, keynotes, and insights, so that users may be more inclined to retweet this information to share it with their followers, rather than replying with individual comments; networking and engaging of attendees and participants with each other, both in person and online, with retweets allowing them to acknowledge and share content from other attendees, speakers, or organizers, contributing to a sense of community and interaction.

Keeping the focus on the days of the conferences, and the week preceding and following, we explore whether retweets and replies provide different *semantic* contributions. More specifically, we are interested in investigating whether retweets and replies are effective proxies for users' agreement and disagreement. To this purpose, we additionally construct a fully-connected **user similarity network** based on the cosine similarity of users' tweets [53]. We group tweets by user and concatenate them after a cleaning procedure involving the normalization of hashtags to terms, the removal of links, and the interpretation of emojis. The resulting texts are encoded using Sentence-BERT [71], a well-known framework for semantic similarity representation based on Transformer models, producing a sentence embedding for each user based on the content of the tweets. We compared the embeddings by cosine-similarity to find sentences with a similar meaning, and built a network of the users with edges weighted according to the similarity score of each user pair.

In the following, we focus on the first and denser COP20 for illustrative purposes, but the following applies to each of the 7 conferences.

We first investigated the user pairs with the highest number of retweets. As shown in Table 5.3, we spotted that 9 out of 10 user pairs have cosine similarity scores

Table 5.3: Top user pairs in COP20 by number of retweets, in descending order of retweet count, and corresponding users statistics. The same symbol in the last column denotes the same user.

# retweets(u,v) + # retweets(v,u)	cos_sim(u,v)	# tweets(u)	# retweets(u,·) + # retweets(·, u)	# tweets(v)	# retweets(v,·) + # retweets(·,v)
611	0.4951	715	711	744	1960*
471	0.4973	47	2708	760	17248**
335	0.5601	364	364	744	1960*
271	0.6684	221	496	760	17248**
189	0.5047	423	1274	760	17248**
184	0.3813	179	690	80	1553
170	0.5858	43	892	760	17248**
151	0.4970	1140	884	168	4843
147	0.4706	43	178	760	17248**
146	0.6960	276	278	301	867

Table 5.4: Subset of replies in disagreement to the original post (first line). The cosine similarity score is wrt the author of the post.

text	cos_sim
Protest image at the Nazca Lines, Peru, today. Full picture	1.0000
HUGE HUGE mistake. Protest image at the Nazca Lines, Peru, today	0.4267
Did you make any damage to Nazca Lines?	0.6954
[…] and stop the bullshit, u just can't protest without putting in danger anything?	0.4720
You had no right to do that in a protected zone. It is disrespectful to our country	0.4420
This is an act of vandalism on a historical landmark not a protest!	0.3937
Walking without authorization in a restricted area!!! WTF	0.5140

greater than or approximately equal to 0.5. The non-direct proportionality between the score and the number of retweets of the pair is due to the users' participation in other conversations, so they share a large piece of text but the remaining is different based on their interactions with other users. On the other hand, we spotted the user pairs with highest cosine similarity scores; we notice that user pairs with cosine similarity tending to 1 have a high number of retweets (typically, ordinary users retweeting authoritative users). We can therefore assert that the retweet interaction corresponds to a form of agreement in our data [20], even at the user-level.

We then looked at the posts with the highest number of replies. We note that although the texts contain clear opposition, some user pairs have high similarity values. This is partly due to the use of similar terms in the debate, but mostly due to the involvement of the users in other discussions: the pairs shown in table 5.4 are involved together in replies of at least one other thread, and some have common retweets. Employing the opposite approach, we investigated the negative scores with higher absolute value and spotted that they do not necessarily correspond to more interactions between the user pair and that negative values never deviate too far from 0; therefore, we cannot assume that replies are a proxy for disagreement at user-level.

Table 5.5: Main structural traits of the retweet networks inferred from the COP data.

	COP20	COP21	COP22	COP23	COP24	COP25	COP26
# Nodes	44 217	324 186	81 864	77 528	73 951	106 274	565 500
# Edges	95 350	818 433	162 447	158 624	136 895	195 070	1 354 656
Density	4e-05	7e-06	2e-05	2e-05	2e-05	1e-05	4e-06
Average In-Degree	2.156	2.525	1.984	2.046	1.851	1.836	2.396
Degree Assortativity	-0.167	-0.072	-0.101	-0.085	-0.093	-0.100	-0.065
% Sources	90.6	89.6	91.0	89.3	90.0	92.4	90.2
% Sinks	3.2	3.6	3.4	3.7	4.4	3.1	3.9
Average Path Length	5.240	5.797	5.572	5.727	6.077	6.287	7.182
Diameter	19	18	16	19	18	19	23
Reciprocity	0.022	0.020	0.021	0.026	0.019	0.015	0.011
Transitivity	0.005	0.002	0.002	0.003	0.002	0.001	0.001
Clustering Coefficient	0.227	0.197	0.234	0.260	0.248	0.224	0.163
Strongly Conn. Comp.	42 482	310 833	79 236	74 306	71 667	103 752	549 133
Weakly Conn. Comp.	296	2378	731	738	953	998	5218
# Communities	2099	16 310	4275	4655	4579	5328	30 998
Modularity	0.547	0.538	0.610	0.606	0.636	0.636	0.664

In light of the above considerations, i.e., accounting that retweets represent the most frequent and informative interaction in the period closest to conferences, we delve deeper into the retweet networks analysis.

5.5.1.1 Retweet network analysis

We analyzed the retweet networks corresponding to each COP through multiple perspectives, aiming at unveiling the main traits that characterize them in terms of network structure and user roles. In this regard, we first characterized the main structural properties of each retweet network from a macroscopic and mesoscopic perspective, as reported in Table 5.5.

Interesting traits emerge by considering the number of nodes and edges within each COP. Indeed, COP21 and COP26 stand out in terms of involved users and relationships compared to others, suggesting greater interactions and engagement around the topics discussed in the corresponding conferences. Further investigations unveiled that such a finding is due to temporal and topical factors, e.g., the *Paris Agreement* (COP20) and the *Glasgow Climate Pact* (COP26).

Users' in-degree, i.e., the number of received retweets, might be leveraged as a proxy for their relevance, or interest; to this aim, we report in Figure 5.2 the Complementary Cumulative Distribution Function of the users' in-degrees across all COPs. While the fraction of relevant users is always quite limited in all scenarios, it turns out that the greater discussion originated from COP21 and COP26 also reflected on users' interest toward some peers, as evidenced by the corresponding tails in Figure 5.2. The impact of users' relevance on link formation (i.e., retweets) was also investigated through the degree assortativity, which shapes how the probability of linkage between users depends on their degrees. We observe it remains negative

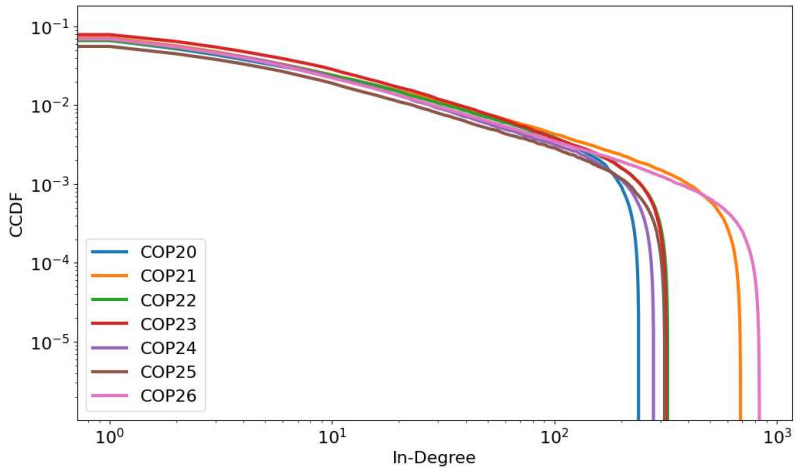


Fig. 5.2: CCDF distribution of the in-degrees of each COP retweet network. Different colors highlight different conferences.

or close to zero in all COPs, indicating a lack of degree correlation among users establishing connections in the network, i.e., users tend to retweet peers based solely on interest in their content rather than on their relevance within the network.

We then investigated the fraction of users who retweet others yet do not receive any retweet (i.e., sources) and vice versa (i.e., sinks), spotting that around 90% of users within each COP do not receive any retweet, whereas a very small fraction never retweeted others' content. Interestingly, such a dichotomy suggests that users generally tend to propagate information produced by a very small fraction of them.

The high values found for path-based measures, such as the average path length and the diameter, indicate the presence of long chains of retweets, denoting potential inefficiency in the network, as it might require several intermediate retweets (and a relatively long time) for a piece of information to propagate from one user to another within the network.

We then delved into link formation (i.e., a proxy for interactions) by investigating how well dyadic and triadic closure principles were met in the networks. As for the former, we spotted a particularly low fraction of reciprocated edges (~2%) for all COPs, hinting at a lack of bi-directionality in interactions. Concerning triadic closure, we delved into both transitivity and local clustering coefficient, spotting in both cases very low values; beyond being expected due to the extremely low density, such values are clues of a reduced engagement and high fragmentation of the networks.

Confirmations for the latter emerge from the remarkable number of strongly and weakly connected components across all COPs, i.e., subgraphs where users can reach with each other by following chains of tweets directly, resp. by ignoring the directionality of the links. We found additional hints at such an interesting

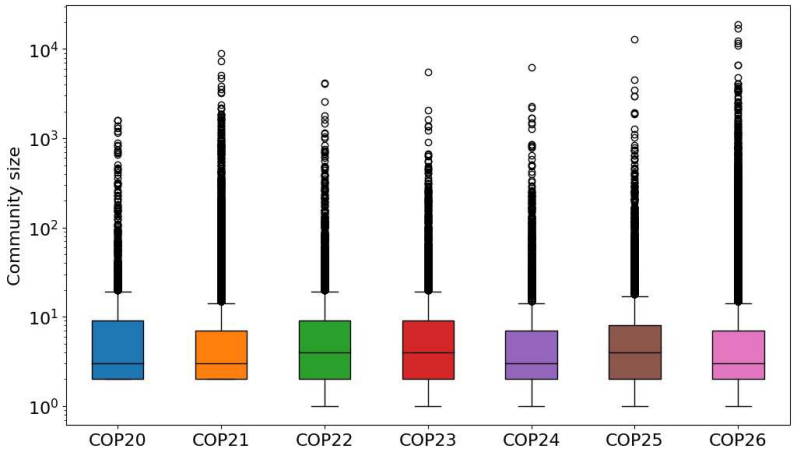


Fig. 5.3: CCDF distribution of community sizes in each COP retweet network. Different colors highlight different conferences.

pattern by looking at the modularity score, i.e., how strongly users are clustered within communities, detected through the widely known Louvain algorithm [3], with internal connectivity way stronger than the external one. Indeed, we spotted that retweet networks are characterized by high modularity, which is also increasing over time, indicating a growing concentration of interactions within specific groups of users, with limited connectivity w.r.t. the remaining users. We hence deepened the structure of such communities for each COP by inspecting their sizes, as reported in Figure 5.3. We noticed that COP21 and COP26 are again those standing out — along with COP25 —, with some larger communities emerging. Although potentially influenced by the broader set of users and links involved, such a finding might be related to higher specialization in specific topics characterizing such COPs, as well as a higher engagement around more relevant figures, as already evidenced in Figure 5.2.

In order to further investigate how user relevance or roles might shape the dynamics within the COPs retweet networks, we shifted our focus toward more local aspects. In this regard, since Twitter used to provide (recently the mechanism for awarding such a label has changed to a paid option), upon assessment of specific requirements, a *verified* label to a very small subset of accredited users, we investigated their distribution in the spotted communities. In particular, we noticed that verified users, who account on average for 3.12% – 6.65% of the community size, follow a distribution in line with the one shown in Figure 5.3.

We then complemented such analysis by looking for the most retweeted users, i.e., influential ones, and those who retweeted the most, i.e., spreader ones. Both user roles are crucial for the robustness of the networks through which information flows, and are strategic when it comes to maximizing the spreading of useful information, or reducing the diffusion of misinformation and fake news. To this aim, we

Table 5.6: Top 5 retweeted (left) and retweeters(right) for each COP. The column count indicates the values of in and out degree, respectively, for each top profile.

	Retweeted		Retweeter	
	Category	Count	Category	Count
COP20	Organization	6243	Activist	589
	Organization	3920	News Spreader	572
	Organization	3909	Activist	537
	Organization	3628	News Spreader	479
	Organization	3017	Activist	465
COP21	Organization	50 026	Activist	2429
	Organization	30 185	Activist	2019
	Official Conf. Account	16 681	Activist	1821
	Organization	15 137	News Spreader	1659
	Organization	14 730	Activist	465
COP22	Organization	17 008	Activist	1027
	Organization	7968	Activist	580
	Official Conf. Account	7180	Activist	487
	Organization	6565	Organization	449
	Organization	3570	Repr. of Organization	443
COP23	Organization	16 385	Activist	508
	Official Conf. Account	7123	Activist	489
	Organization	2987	Activist	395
	Organization	2330	Repr. of Organization	349
	Actor and ex-Politician	1745	Repr. of Organization	342
COP24	Organization	17 316	Activist	668
	Repr. of Organization	5106	News Spreader	564
	Activist	4791	Activist	393
	Organization	3754	Academic	310
	Action Movement	3566	Academic/Repr. of Org.	310
COP25	Activist	21 874	Activist	737
	Organization	10 233	Activist	726
	Repr. of Organization	9747	Academic/Repr. of Org.	409
	Academic	5964	Superuser	315
	Politician	4542	Superuser	300
COP26	Official Conf. Account	57 216	Superuser	1955
	Activist	48 373	Artist	1942
	Band	20 443	Activist	1312
	Journalist	18 209	Organization	1132
	News Spreader	12 577	Activist	1132

report in Table 5.6 the top-5 users in descending order of in-degree and out-degree, respectively.

As it can be noticed from Table 5.6 (left), the most retweeted accounts in earlier COPs primarily correspond to international organizations involved in climate-related issues, as well as official conference accounts. However, starting from COP23, a greater interest in such issues determined the engagement of individual personalities in the social debate, lending their support to the cause; as a result, the most retweeted accounts started including public figures (e.g., politicians or journalists).

As concerns top spreaders, Table 5.6 (right) shows the predominance of activists, as expected given their dedication to the cause, as well as representatives from organi-

zations and news spreaders (e.g., online newspapers or blogs). Interestingly, not all of them are recognized as verified. Besides, we surprisingly spotted that top retweeted and retweeters are likely to share the same communities. Furthermore, recent COPs see some very active emerging profiles on the issue despite not being explicitly stated as activists, an interesting evidence of how these issues are increasingly embedded in the social debate and not the prerogative of a few.

5.5.2 Topic modeling

Given the vast amount of data generated while discussing climate change and COPs, we leveraged BERTopic to extract relevant and meaningful topics. Such analysis allowed us to identify and understand the key issues and trends discussed, as well as detect potential shifts in topics that characterized seven years of climate debate. We report the top-5 (by volume of tweets) discussed topics for each COP in Table 5.7.

As expected, some inherent topics are commonly shared by different COPs, reflecting the collective recognition of the urgency to address climate change. We then spotted the topic of deforestation, which consistently emerges from different meetings (COP20, COP21) as an indication of the awareness of their impact on climate change. Similarly, the topic of agriculture spans multiple COPs (i.e., from COP21 to COP23), underlying its essential role in both contributing to and being impacted by climate change. Renewable energies stem from among the most discussed topics in COP21 and COP22, emphasizing the need to transition to more sustainable sources of energy. Similarly, the need for increased funding and financial resources to address climate change characterizes COP21 and COP23, suggesting the need for enhanced efforts. Interestingly, more recent COPs (i.e., COP24 and COP25) are characterized by an emerging discussion around youth engagement and activism in the climate cause, indicating a growing recognition of the importance of youth in driving climate action. Besides, we noticed an increased demand for better information (from COP23 to COP25), as a tool for increasing awareness and bearing to broader climate action. Finally, the latest COPs are characterized by issue-specific discussions, such as the role of science and the influence of politics in climate issues (COP25), or the demand for reducing aviation emissions and making the cryptocurrency ecosystem more sustainable (COP26).

5.5.3 Affective computing

In order to analyze the variation in public opinion, we divided our corpus of tweets of each COP into three splits: (i) one week before the event, (ii) the dates of the event, (iii) one week after the event. Such a subdivision, in fact, allowed us to assess any pre-COP expectation, as well as immediate reactions due to decisions made during the events.

Table 5.7: Main topics discussed in each COP

	Main Topics
COP20	Impact of human activities and efforts to address climate change International climate negotiations, agreements, and commitments Urgent need for action to protect the Earth and save the planet Need for global cooperation and sustainable solutions Impact of deforestation and reforestation on ecosystems and landscapes
COP21	Negotiations and progress surrounding the Paris Agreement Ensuring justice and equity, mobilize financial resources Renewable energy, with focus on solar energy Agriculture, sustainable practices and agricultural policies (Rain)forest preservation, impact of deforestation, and need for reforestation
COP22	Climate hope, sustainability, eco-friendly living, connection to political events Carbon markets and economic aspects of addressing climate change Renewable energy, clean energy initiatives and role of regulatory agencies Participation, activities, and events associated with the conference Sustainable agriculture, climate-smart practices, and ecological aspects
COP23	Promoting climate justice, and mobilizing financial resources Dissemination of news and information related to ecological concerns Agriculture, sustainable practices, and role of farmers Involvement of volunteers, events, partnerships, and a general enthusiasm Challenges faced by Pacific Island nations and their efforts
COP24	International negotiations and the involvement of youth in climate activism Regular updates about sustainability and environmental concerns Clean energy, collective efforts and anticipation for future initiatives Climate change, extinction, and the role of human civilization News, reporting, updates, and events related to the conference.
COP25	Urgency for action, sustainable practices and emissions reduction Media coverage and youth engagement in climate activism Disappointment with the outcomes, and assessment of progress made by countries Role of science, assessments of success or failure, and political influence Climate leaders, and the importance of taking immediate action for a better future
COP26	Need for sustainable practices and collaborative efforts Severity of the climate situation, climate justice and emissions reduction Reducing aviation emissions and promoting sustainable aviation practices Intersection of sustainability efforts with cryptocurrency and related digital assets Inclusion and rights of people with disabilities

Using the VADER sentiment analysis tool, we assigned a score $s \in [-1, 1]$ to each tweet of the corresponding time slice. We then used a threshold-based categorization to label each tweet as follows: *extremely negative* ($s \leq -0.75$), *negative* ($s \in (-0.75, -0.15)$), *neutral* ($s \in [-0.15, 0.15]$), *positive* ($s \in (0.15, 0.75)$), and *extremely positive* ($s \geq -0.75$). Figure 5.4 illustrates the percentage values of sentiment, categorized according to the five classes, for each time slice of each COP.
The plot reveals a certain temporal consistency of the neutral component over time, as well as during the time slices that characterize each COP. Extremely negative and positive components account for a reduced fraction of tweets, although they provide valuable insights. As for negative components, we noticed spikes — also in terms of extreme ones — during the latest COPs, which may suggest increasing awareness

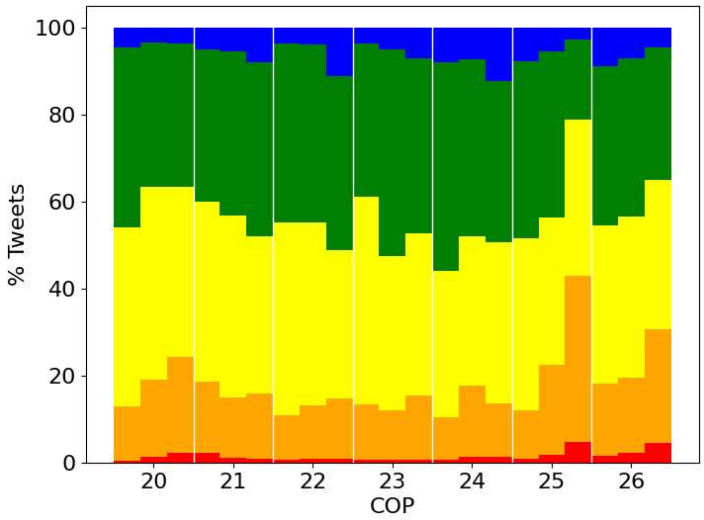


Fig. 5.4: Stacked barchart displaying the percentage of tweets, for each time slice of each COP, belonging to the 5 sentiment classes: extr. positive (blue), positive (green), neutral (yellow), negative (orange) and extr. negative (red).

of the climate crisis and the catastrophic effects it may have (and is already having) on the planet and our society. Besides, negativity tends to increase right after most conferences, suggesting that they actually stir things up. An upward trend is observed for post-conference positive components between COP21 and COP24, which also generated some spikes in extremely positive ones. Overall, we spotted that positive sentiments account for a higher proportion than negative ones, hinting at trust mechanisms w.r.t. events in which the climate crisis is addressed by prominent personalities with decision-making power. Nonetheless, such an observed positive trend reverses starting from COP25, achieving the highest degrees of negativity and the lowest of positivity, including extreme components. This anomaly is all but unexpected, as COP25 represents the most protracted and challenging climate conference in recent years, characterized by the inability to achieve a binding agreement and to fulfill the Paris Agreement for the subsequent year, coupled with a new record in levels of greenhouse gases — impeding heat dissipation within the atmosphere.

We gained a deeper understanding of such findings by extracting emotions conveyed by discussions around the climate debate, through EmoAtlas and Plutchik's wheel visualization. Specifically, we want to characterize the extremely negative and positive debate for the three observation periods related to each COP. For instance, Figure 5.5-left illustrates the breakdown of the extremely negative sentiment corresponding to the post-COP25 period, showing *anger*, *disgust* and *sadness* as predominant emotions; by contrast, Figure 5.5-right shows that the pre-COP26 period is characterized by an extremely positive sentiment, given the high scores associated

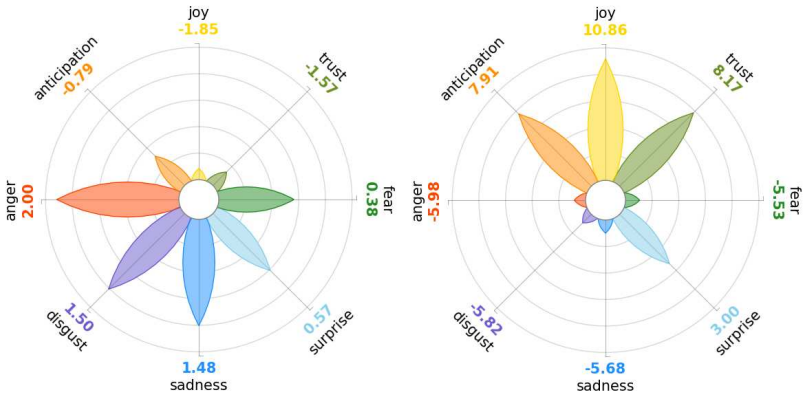


Fig. 5.5: Plutchik's wheel of emotions displaying the 8 basic emotions for the extr. negative component of sentiment after COP25 (left) and for the extr. positive component of sentiment before COP26 (right).

with *joy*, *trust* and *anticipation*. Notably, for the extremely positive components, we point out the high values of *joy* compared to the other scores, as it exhibits peaks during the conference periods when the social debate is most intense, whereas *surprise* decreases in the period following the conferences; also, *anticipation* increases during the conference period compared to the previous week but decreases in the following week, with a significant drop observed after COP23. Consistently higher values of *trust* are reported after the conferences, except for COP26, due to exceptionally high initial values. For the extremely negative component, *anger* and *fear* consistently decrease in the period following the conferences. COP26 is also the only event to exhibit decreasing values of *disgust* across the three periods.

5.6 Concluding remarks

In this chapter, we contributed to the understanding of the social debate on climate change by exploiting the discussion surrounding seven years of the Conference of the Parties on Twitter as a proxy for the physical perception of the issue. Graph mining and text mining techniques have been proven helpful tools for exploiting the social traces people left on social platforms, providing valuable insights that might be used to make better decisions, prioritize actions, and foster more effective communication for the urgent challenge of climate change. The proposed modular approach promotes clearer insights into different aspects of the data, such as the structure of interactions, thematic patterns in discussions, and emotional tone of conversations. In the future, a multidimensional analysis that combines the results from each module can provide a holistic understanding of online community dynamics and underlying phenomena,

including how social structures influence discussion topics and emotional expressions. Future work may also include the use of real-time data that further benefit from people's proven ability to act as social sensors in the way they react to emergent phenomena on social, as well as the integration of additional techniques and data sources to provide a comprehensive overview of the climate change perception.

Some open Large Language Models for analyzing agreement and disagreement between pairs of users from their responses are currently under investigation [53]. The advantage of such models lies in their ability to understand context and capture the nuances of language, including sarcasm, irony, or joke, as well as criticism, skepticism, or irrelevant questioning, but they also come with limitations, primarily the need for human oversight to ensure the accuracy and fairness of the outcomes.

Chapter 6
Conclusions

This work focuses on the integration of multiple feature-rich network models to provide a holistic and nuanced understanding of the underlying data, addressing the prominent challenge of encapsulating the complexity of real-world systems — inherently interconnected and feature-rich — within a unique network model without sacrificing information. The common goal is to facilitate more effective analysis, interpretation, and utilization of interconnected data in diverse domains, thereby unlocking new frontiers in network science and beyond. More specifically, three main contributions can be identified, as detailed below.

Contrastive learning framework for multilayer heterogeneous attributed networks. Chapter 3 proposes a contrastive graph representation learning framework, named Co-MLHAN, for networks that are simultaneously heterogeneous, attributed and multilayer, i.e., characterized by multiple types of nodes and relationships, additional information content associated to nodes, and replication of nodes on multiple layers. Remarkably, our framework demonstrates proficiency in handling networks where each layer is a heterogeneous graph with attributed nodes, as well as accommodating both intra- and inter-layer connections between nodes. The proposed framework learns multi-type node embeddings in an unsupervised setting, i.e., without relying on labeled data, based on a cooperative contrastive mechanism between pairs of node representations learned via hierarchical GNN modules under two views of the original graph, the *network schema view* and the *meta-path view*, able to encode local and high-order structures of nodes, respectively. We devise two alternative strategies to encode the across-layer information underlying replicas of nodes, i.e., *pillar edges*, on which basis we propose two methods, or variants of the framework: Co-MLHAN models a novel variant of meta-path, which we named *across-layer meta-path*, with terminal nodes belonging to different layers and the intermediate node matching a pillar-edge, while Co-MLHAN-SA directly considers all the instances of the same entities in multiple layers via a *Supra-Adjacency matrix*.

The learned representations are task-independent, and any of the two optimized embeddings can be used for different downstream graph mining tasks, both at entity/node level, edge level or graph level. Experimental results on the entity (multi-

class) classification task showed that the proposed methods significantly outperforme existing competitors for multi-type node embeddings in an unsupervised setting, effectively exploiting external content and multilayer information, and achieved similar performance in all the experiments, showing that both approaches are resilient to the selection strategy of positive examples, and can successfully embed attributes at node/entity level and node-tailored feature information.

Continual learning framework for dynamic heterogeneous attributed networks. Chapter 4 proposes a continual graph representation learning framework, named DyHANE, for networks that are simultaneously heterogeneous, attributed and dynamic, i.e., characterized by multiple types of nodes and relationships, additional information content associated to nodes, changes in network topology (newly formed or deleted nodes and/or relationships) and modified node features over time. The core of DyHANE is an incremental GNN module, initialized with parameters learned at the previous timestamp and trained on a subset of changed and unchanged nodes, capable of learning effective and up-to-date representations for all nodes within the network. Remarkably, at each new timestamp, it identifies an informative sample of multi-typed nodes as training set, comprising both nodes affected by changes, i.e., *knew knowledge*, and most relevant unchanged nodes, i.e., *previous knowledge*, stored as experience. We propose two algorithms, one for detecting a reduced set of nodes affected by changes based on a novel variant of meta-path, which we named *incremental meta-path*, and the other for detecting a reduced set of nodes to be used as experience node replay, based on a *GNN-Explainer*.

On a multi-step, data incremental (multi-class) classification task, we demonstrate the advantages of modeling offered by heterogeneous networks and show the ability of our framework to achieve good performance on both changed and unchanged nodes, assessing the effectiveness of our method to learn new patterns and to retain unchanged patterns while integrating new ones. DyHANE strikes a satisfying balance between the performance achieved by *retrained approaches*, i.e., same models retrained from scratch at each timestamp, in terms of accuracy in classifying changed and unchanged nodes, and by *online approaches*, i.e., same models trained solely on changed nodes, in terms of efficient incorporation of new knowledge.

Complex social networks analysis. Chapter 5 proposes a combination of graph mining and text mining techniques to analyze the evolution of social debate on a thematically focused content — ensuring more intense interactions — and featured as a discrete, regularly repeated online event for multi-year analysis. By exploiting the discussion on climate crisis surrounding seven years of the Conference of the Parties on Twitter as a proxy for the physical perception of the issue, we propose a modular approach comprising three main modules for network analysis, topic modeling, and affective computing, to supports a systematic, specialized, and comprehensive analysis of complex social data. More specifically, we infer graph-based models to characterize the main framework underlying the social interactions and shed light on the most influential users for each COP; we leverage topic modeling to extract the most discussed topics for each conference; we use lexicon-based frameworks

for shaping polarity and emotions expressed by users via textual components, to investigate how they evolve over time.

We construct a multilayer *user interaction network* modeling *retweets* and *replies* as direct interactions between user pairs, whose combination with a flattened fully-connected *user similarity network* based on the cosine similarity of users' contents demonstrates that retweet interactions, in our data, correspond to a form of agreement at user-level. The extracted *retweet network* showed to be effective in unveiling the actors that drive the debate, deciphering information amplification patterns and addressing misinformation and disinformation challenges tracing origins and understanding propagation mechanisms. We further investigate the context of debates, performing temporal analysis to figure out topic evolution and key events shaping discussions, as well as assessing sentiment and emotions sorrounding the social debate. For a more in-depth analysis, we split the data used in the affective computing module into three parts: the conferences days, the week preeceding and the week following, to assess any pre-COP expectation or immediate reactions due to decision made during the events. Remarkably, the separation of the analysis into distinct modules enables specialized focus within each domain, fostering more effective and efficient analysis tailored to specific tasks, and enhances flexibility and scalability, facilitating easier modification or updating of individual components without disrupting the entire system.

Future directions

The flexibility of the proposed approaches enables further optimization of stages and modules. Specifically, we are interested in investigating different strategies for the selection of positive and/or negative examples in the contrastive learning framework, e.g., by exploiting the learned node features in addition to structural information; we want to examine new algorithms for detecting nodes affected by changes and experience nodes on heterogeneous graphs of growing size,i.e., by exploring state-of-the-art active learning techniques; we are also curious about a multidimensional analysis that combines the results from each module of the social network analysis to provide a holistic understanding of the social debate, e.g., addressing how social structures influence discussion topics and emotional expressions. Furthermore, we would like to enrich social network models with multiple node types (e.g., posts and hashtags) to analyze their contributions in shedding light on the nuanced interplay between social structures, thematic content, and user sentiment surrounding the debate. It will also be interesting to stress the flexibility of the proposed approaches on new datasets and applications with unseen properties, as well as on multiple downstream graph learning tasks.

The results achieved so far will be the foundation for generating fresh perspectives and ideas. As stated by Albert Szent-Györgyi, a Hungarian biochemist and winner of the Nobel Prize in Physiology or Medicine in 1937, *"Research is seeing what everybody else has seen and thinking what nobody else has thought"*.

Glossary

Attributed Network . Graph characterized by external information content associated with nodes and/or relationships in the form of attribute vectors.

Contrastive learning . Unsupervised machine learning paradigm where the model is trained to distinguish between similar and dissimilar examples (nodes or subgraphs) in the dataset. The goal is to learn representations of the data so that similar examples are close together in the feature space, while dissimilar examples are far apart.

Dynamic Network . Graph characterized by changes over time in network topology and node/edge attributes.

Embedding . Dense vector representation encoding graph-structured data in a lower dimensional space.

Feature-rich Network . Network model where the expressive power of the graph topology is enhanced by exposing one or more peculiar features.

Graph Representation Learning . Process of automatically extracting meaningful and concise representations (embeddings) from graph-structured data.

Heterogeneous (Information) Network Graph characterized by multiple entity and/or relationship types.

Memory buffer (or experience nodes) . Set of most representative unchanged nodes that are stored as experience and replayed at following timestamps in replay-based methods on dynamic networks.

Meta-path Composite relation used in Heterogeneous Information Networks to model high-order proximity between two reachable nodes not directly connected in the graph.

Multilayer Network . Graph characterized by replicas of nodes on multiple layers according to different point of views, properties, temporal intervals or other semantics aspects.

Network schema . Abstraction of the Heterogeneous Information Network showing the different node and relationship types.

Replay-based (or rehearsal) method . Method that involves storing representative historical data and incorporating it during the training of new tasks to prevent the loss of knowledge.

View of the graph . Perspective or subset of the original graph, in analogy to relational databases. It is created by defining a query or traversal pattern that specifies which nodes, relationships, and properties are included in the view.

References

1. Ahrabian, K., Feizi, A., Salehi, Y., Hamilton, W.L., Bose, A.J.: Structure aware negative sampling in knowledge graphs. CoRR **abs/2009.11355** (2020)
2. Baevski, A., Hsu, W.N., Xu, Q., Babu, A., Gu, J., Auli, M.: data2vec: A general framework for self-supervised learning in speech, vision and language (2022). DOI 10.48550/ARXIV.2202.03555
3. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment **2008**(10), P10008 (2008). DOI 10.1088/1742-5468/2008/10/P10008
4. Brody, S., Alon, U., Yahav, E.: How attentive are graph attention networks? CoRR **abs/2105.14491** (2021)
5. Cen, Y., Zou, X., Zhang, J., Yang, H., Zhou, J., Tang, J.: Representation learning for attributed multiplex heterogeneous network. CoRR **abs/1905.01669** (2019)
6. Chen, F., Wang, Y., Wang, B., Kuo, C.J.: Graph representation learning: A survey. CoRR **abs/1909.00958** (2019)
7. Chen, J., Ma, T., Xiao, C.: Fastgcn: Fast learning with graph convolutional networks via importance sampling. CoRR **abs/1801.10247** (2018)
8. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations (2020). DOI 10.48550/ARXIV.2002.05709
9. Chen, Z., Shu, J., Liu, L.: The node importance evaluation method based on graph convolution in multilayer heterogeneous networks. Connect. Sci. **35**(1) (2023). DOI 10.1080/09540091.2023.2229964
10. Cody, E.M., Reagan, A.J., Mitchell, L., Dodds, P.S., Danforth, C.M.: Climate change sentiment on twitter: An unsolicited public opinion poll. PloS one **10**(8), e0136092 (2015)
11. Coscia, M.: The atlas for the aspiring network scientist. CoRR **abs/2101.00863** (2021)
12. Dahal, B., Kumar, S.A., Li, Z.: Topic modeling and sentiment analysis of global climate change tweets. Social network analysis and mining **9**, 1–20 (2019)
13. Dong, Y., Chawla, N.V., Swami, A.: metapath2vec: Scalable representation learning for heterogeneous networks. In: In Proc. of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 135–144. ACM (2017). DOI 10.1145/3097983.3098036
14. Du, L., Wang, Y., Song, G., Lu, Z., Wang, J.: Dynamic network embedding : An extended approach for skip-gram based network embedding. In: In Proc. of the 27th International Joint Conference on Artificial Intelligence (IJCAI), pp. 2086–2092. ijcai.org (2018). DOI 10.24963/IJCAI.2018/288
15. Dunlap, R.E., McCright, A.M., Yarosh, J.H.: The political divide on climate change: Partisan polarization widens in the us. Environment: Science and Policy for Sustainable Development **58**(5), 4–23 (2016)
16. Dwivedi, V.P., Bresson, X.: A generalization of transformer networks to graphs. CoRR **abs/2012.09699** (2020)
17. Effrosynidis, D., Sylaios, G., Arampatzis, A.: Exploring climate change on twitter using seven aspects: Stance, sentiment, aggressiveness, temperature, gender, topics, and disasters. Plos one **17**(9), e0274213 (2022)
18. Falkenberg, M., Galeazzi, A., Torricelli, M., Di Marco, N., Larosa, F., Sas, M., Mekacher, A., Pearce, W., Zollo, F., Quattrociocchi, W., Baronchelli, A.: Growing polarization around climate change on social media. Nature Climate Change **12**(12), 1114–1121 (2022). DOI 10.1038/s41558-022-01527-x
19. Fu, X., Zhang, J., Meng, Z., King, I.: MAGNN: metapath aggregated graph neural network for heterogeneous graph embedding. CoRR **abs/2002.01680** (2020)
20. Gaumont, N., Panahi, M., Chavalarias, D.: Reconstruction of the socio-semantic dynamics of political activist twitter networks—method and application to the 2017 french presidential election. PloS one **13**(9), e0201879 (2018)

21. Ghorbani, M., Baghshah, M.S., Rabiee, H.R.: MGCN: semi-supervised classification in multi-layer graphs with graph convolutional networks. In: In Proc. of the 11th International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 208–211. ACM (2019). DOI 10.1145/3341161.3342942
22. Grassia, M., Domenico, M.D., Mangioni, G.: mgnn: Generalizing the graph neural networks to the multilayer case. CoRR **abs/2109.10119** (2021)
23. Grootendorst, M.: Bertopic: Neural topic modeling with a class-based TF-IDF procedure. CoRR **abs/2203.05794** (2022). DOI 10.48550/arXiv.2203.05794
24. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. CoRR **abs/1706.02216** (2017)
25. Hassani, K., Ahmadi, A.H.K.: Contrastive multi-view representation learning on graphs. CoRR **abs/2006.05582** (2020)
26. Hoang, V.T., Jeon, H., You, E., Yoon, Y., Jung, S., Lee, O.: Graph representation learning and its applications: A survey. Sensors **23**(8), 4168 (2023). DOI 10.3390/S23084168
27. Hu, Z., Dong, Y., Wang, K., Sun, Y.: Heterogeneous graph transformer. CoRR **abs/2003.01332** (2020)
28. Hutto, C.J., Gilbert, E.: VADER: A parsimonious rule-based model for sentiment analysis of social media text. In: In Proc. of the 8th International Conference on Weblogs and Social Media (ICWSM) (2014)
29. Interdonato, R., Atzmueller, M., Gaito, S., Kanawati, R., Largeron, C., Sala, A.: Feature-rich networks: going beyond complex network topologies. Appl. Netw. Sci. **4**(1), 4:1–4:13 (2019). DOI 10.1007/S41109-019-0111-X
30. Jang, S.M., Hart, P.S.: Polarized frames on "climate change" and "global warming" across countries and states: Evidence from twitter big data. Global environmental change **32**, 11–17 (2015)
31. Jin, Z., Wang, Y., Wang, Q., Ming, Y., Ma, T., Qu, H.: Gnnlens: A visual analytics approach for prediction error diagnosis of graph neural networks. IEEE Transactions on Visualization and Computer Graphics (2022)
32. Jing, B., Xiang, Y., Chen, X., Chen, Y., Tong, H.: Graph-mvp: Multi-view prototypical contrastive learning for multiplex graphs. CoRR **abs/2109.03560** (2021)
33. Ju, W., Fang, Z., Gu, Y., Liu, Z., Long, Q., Qiao, Z., Qin, Y., Shen, J., Sun, F., Xiao, Z., Yang, J., Yuan, J., Zhao, Y., Luo, X., Zhang, M.: A comprehensive survey on deep graph representation learning. CoRR **abs/2304.05055** (2023). DOI 10.48550/ARXIV.2304.05055
34. Kalantidis, Y., Sariyildiz, M.B., Pion, N., Weinzaepfel, P., Larlus, D.: Hard negative mixing for contrastive learning. CoRR **abs/2010.01028** (2020)
35. Khan, R.A., Kleinsteuber, M.: A framework for joint unsupervised learning of cluster-aware embedding for heterogeneous networks. CoRR **abs/2108.03953** (2021)
36. Khoshraftar, S., An, A.: A survey on graph representation learning methods. CoRR **abs/2204.01855** (2022). DOI 10.48550/arXiv.2204.01855
37. Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., Krishnan, D.: Supervised contrastive learning. CoRR **abs/2004.11362** (2020)
38. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2017)
39. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: In Proc. of the 5th International Conference on Learning Representations (ICLR) (2017)
40. Kirilenko, A.P., Molodtsova, T., Stepchenkova, S.O.: People as sensors: Mass media and local temperature influence climate change discussion on twitter. Global Environmental Change **30**, 92–100 (2015)
41. Kirkpatrick, J., Pascanu, R., Rabinowitz, N.C., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., Hadsell, R.: Overcoming catastrophic forgetting in neural networks. CoRR **abs/1612.00796** (2016)
42. Li, G., Muller, M., Thabet, A., Ghanem, B.: Deepgcns: Can gcns go as deep as cnns? In: In Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV) (2019)

43. Li, H., Wang, Y., Lyu, Z., Shi, J.: Multi-task learning for recommendation over heterogeneous information network. IEEE Transactions on Knowledge and Data Engineering (2020)
44. Lin, B., Wang, X., Dong, Y., Huo, C., Ren, W., Xu, C.: Metapaths guided neighbors aggregated network for heterogeneous graph reasoning (2021). DOI 10.48550/ARXIV.2103.06474
45. Linsker, R.: Self-organization in a perceptual network. Computer **21**(3), 105–117 (1988). DOI 10.1109/2.36
46. Liu, L., Kang, Z., Tian, L., Xu, W., He, X.: Multilayer graph contrastive clustering network. CoRR **abs/2112.14021** (2021)
47. Liu, Y., Pan, S., Jin, M., Zhou, C., Xia, F., Yu, P.S.: Graph self-supervised learning: A survey. CoRR **abs/2103.00111** (2021)
48. Ma, Y., Guo, Z., Ren, Z., Tang, J., Yin, D.: Streaming graph neural networks. In: In Proc. of the 43rd International ACM SIGIR conference on research and development in Information Retrieval (SIGIR), pp. 719–728. ACM (2020). DOI 10.1145/3397271.3401092
49. Ma, Y., Wang, S., Aggarwal, C.C., Yin, D., Tang, J.: Multi-dimensional graph convolutional networks. In: In Proc. of the SIAM International Conference on Data Mining (SDM), pp. 657–665. SIAM (2019)
50. van der Maaten, L., Hinton, G.: Visualizing Data Using t-SNE. Journal of Machine Learning Research **9**, 2579–2605 (2008)
51. Manchanda, S., Zheng, D., Karypis, G.: Schema-aware deep graph convolutional networks for heterogeneous graphs. CoRR **abs/2105.00644** (2021)
52. Martirano, L., Ienco, D., Interdonato, R., Tagarelli, A.: Dyhane: Dynamic heterogeneous attributed network embedding. Appl. Netw. Sci. (Under Review)
53. Martirano, L., La Cava, L., Gullo, F., Tagarelli, A.: Agreement and disagreement on climate crisis: insights from Twitter during the Conferences of the Parties. In: In Proc. of the 12th International Conference on Complex Networks and their Applications (CNA) – Book of abstracts, pp. 108–111 (2023)
54. Martirano, L., La Cava, L., Tagarelli, A.: Evolution of the social debate on climate crisis: Insights from twitter during the conferences of the parties. In: In Proc. of the 2023 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM), pp. 1–6. IEEE (2023)
55. Martirano, L., Zangari, L., Tagarelli, A.: Co-mlhan: Contrastive learning for multilayer heterogeneous attributed networks. Applied Network Science **7**(1), 65 (2022). DOI 10.1007/S41109-022-00504-9
56. Mavromatis, C., Karypis, G.: Hemi: Multi-view embedding in heterogeneous graphs. CoRR **abs/2109.07008** (2021)
57. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. In: Psychology of learning and motivation, vol. 24, pp. 109–165. Elsevier (1989). DOI https://doi.org/10.1016/S0079-7421(08)60536-8
58. McInnes, L., Healy, J., Melville, J.: Umap: Uniform manifold approximation and projection for dimension reduction (2018). DOI 10.48550/ARXIV.1802.03426
59. Mendoza, M., Poblete, B., Valderrama, I.: Nowcasting earthquake damages with twitter. EPJ Data Science **8**(1), 1–23 (2019)
60. Mukunda, K., Roy, A., Santra, A., Chakravarthy, S.: Stress centrality in heterogeneous multilayer networks: Heuristics-based detection. In: In Proc. of the 9th International Conference on Big Data Computing Service and Applications (BigDataService), pp. 103–110. IEEE (2023). DOI 10.1109/BIGDATASERVICE58306.2023.00021
61. Mukunda, K., Santra, A., Chakravarthy, S.: The challenge of finding degree centrality nodes in heterogeneous multilayer networks. In: In Proc. of the 31st Symposium of Advanced Database Systems (SEBD), *CEUR Workshop Proceedings*, vol. 3478, pp. 335–348. CEUR-WS.org (2023)
62. Nasrazadani, M., Fatemi, A., Nematbakhsh, M.: Sign prediction in sparse social networks using clustering and collaborative filtering. J. Supercomput. **78**(1), 596–615 (2022)
63. Park, C., Kim, D., Han, J., Yu, H.: Unsupervised attributed multiplex network embedding. CoRR **abs/1911.06750** (2019)

64. Pearce, W., Holmberg, K., Hellsten, I., Nerlich, B.: Climate change on twitter: Topics, communities and conversations about the 2013 ipcc working group 1 report. PloS one **9**(4), e94785 (2014)
65. Pearson, K.: Liii. on lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin philosophical magazine and journal of science **2**(11), 559–572 (1901)
66. Peng, H., Yang, R., Wang, Z., Li, J., He, L., Yu, P.S., Zomaya, A.Y., Ranjan, R.: Lime: Low-cost and incremental learning for dynamic heterogeneous information networks. IEEE Trans. Computers **71**(3), 628–642 (2022). DOI 10.1109/TC.2021.3057082
67. Perini, M., Ramponi, G., Carbone, P., Kalavri, V.: Learning on streaming graphs with experience replay. In: J. Hong, M. Bures, J.W. Park, T. Cerný (eds.) In Proc. of the 37th ACM/SIGAPP Symposium on Applied Computing, pp. 470–478. ACM (2022). DOI 10.1145/3477314.3507113
68. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: In Proc. of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710 (2014)
69. Plutchik, R.: Emotions: A general psychoevolutionary theory. Approaches to emotion **1984**(197-219), 2–4 (1984)
70. Priem, J., Piwowar, H.A., Orr, R.: Openalex: A fully-open index of scholarly works, authors, venues, institutions, and concepts. CoRR **abs/2205.01833** (2022). DOI 10.48550/ARXIV.2205.01833
71. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. CoRR **abs/1908.10084** (2019)
72. Robinson, B., Power, R., Cameron, M.: A sensitive twitter earthquake detector. In: In Proc. of the 22nd ACM Conference on World Wide Web (WWW), p. 999–1002 (2013). DOI 10.1145/2487788.2488101
73. Robinson, J., Chuang, C., Sra, S., Jegelka, S.: Contrastive learning with hard negative samples. CoRR **abs/2010.04592** (2020)
74. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes twitter users: Real-time event detection by social sensors. In: In Proc. of the 19th ACM Conference on World Wide Web (WWW), p. 851–860 (2010). DOI 10.1145/1772690.1772777
75. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Networks **20**(1), 61–80 (2009). DOI 10.1109/TNN.2008.2005605
76. Shang, J., Qu, M., Liu, J., Kaplan, L.M., Han, J., Peng, J.: Meta-path guided embedding for similarity search in large-scale heterogeneous information networks. CoRR **abs/1610.09769** (2016)
77. Shanthamallu, U.S., Thiagarajan, J.J., Song, H., Spanias, A.: GrAMME: Semisupervised Learning Using Multilayered Graph Attention Models. IEEE Transactions on Neural Networks and Learning Systems **31**(10), 3977–3988 (2020). DOI 10.1109/TNNLS.2019.2948797
78. Shen, Z., Ma, H., Wang, K.: A web-scale system for scientific knowledge exploration. CoRR **abs/1805.12216** (2018)
79. Shi, B., Weninger, T.: Mining interesting meta-paths from complex heterogeneous information networks. In: In Proc. of the 2014 IEEE International Conference on Data Mining Workshop, pp. 488–495. IEEE (2014)
80. Shi, S., Xie, P., Luo, X., Qiao, K., Wang, L., Chen, J., Yan, B.: Adaptive multi-layer contrastive graph neural networks. CoRR **abs/2109.14159** (2021)
81. Stella, M., De Nigris, S., Aloric, A., Siew, C.S.: Forma mentis networks quantify crucial differences in stem perception between students and experts. PloS one **14**(10), e0222870 (2019)
82. Sun, Y., Han, J.: Mining Heterogeneous Information Networks: Principles and Methodologies. Synthesis Lectures on Data Mining and Knowledge Discovery. Morgan & Claypool Publishers (2012). DOI 10.2200/S00433ED1V01Y201207DMK005
83. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. CoRR **abs/1703.01365** (2017)

84. Trivedi, R., Farajtabar, M., Biswal, P., Zha, H.: Dyrep: Learning representations over dynamic graphs. In: In Proc. of the 7th International Conference on Learning Representations (ICLR). OpenReview.net (2019)
85. Tyagi, A., Uyheng, J., Carley, K.M.: Affective polarization in online climate change discourse on twitter. In: In Proc. of the 12th IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 443–447 (2020). DOI 10.1109/ASONAM49781.2020.9381419
86. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: In Proc. of the 31st Annual Conference on Advances in Neural Information Processing Systems (NeurIPS), vol. 30 (2017)
87. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: In Proc. of the 6th International Conference on Learning Representations (ICLR) (2018)
88. Wan, G., Du, B., Pan, S., Haffari, G.: Reinforcement learning based meta-path discovery in large-scale heterogeneous information networks. In: In Proc. of the aaai conference on artificial intelligence, vol. 34, pp. 6094–6101 (2020)
89. Wang, J., Song, G., Wu, Y., Wang, L.: Streaming graph neural networks via continual learning. CoRR **abs/2009.10951** (2020)
90. Wang, J., Zhu, W., Song, G., Wang, L.: Streaming graph neural networks with generative replay. In: In Proc. of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 1878–1888. ACM (2022). DOI 10.1145/3534678.3539336
91. Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., Xiao, T., He, T., Karypis, G., Li, J., Zhang, Z.: Deep graph library: A graph-centric, highly-performant package for graph neural networks. CoRR **abs/1909.01315** (2020)
92. Wang, X., Ji, H., Shi, C., Wang, B., Cui, P., Yu, P.S., Ye, Y.: Heterogeneous graph attention network. CoRR **abs/1903.07293** (2019)
93. Wang, X., Liu, N., Han, H., Shi, C.: Self-supervised heterogeneous graph neural network with co-contrastive learning. CoRR **abs/2105.09111** (2021)
94. Wasserman, S., Faust, K.: Social network analysis: Methods and applications. Cambridge university press (1994)
95. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world'networks. nature **393**(6684), 440–442 (1998)
96. Wei, X., Liu, Z., Sun, L., Yu, P.S.: Unsupervised meta-path reduction on heterogeneous information networks. arXiv preprint arXiv:1810.12503 (2018)
97. Williams, H.T., McMurray, J.R., Kurz, T., Lambert, F.H.: Network analysis reveals open forums and echo chambers in social media discussions of climate change. Global environmental change **32**, 126–138 (2015)
98. Xie, Y., Ou, Z., Chen, L., Liu, Y., Xu, K., Yang, C., Zheng, Z.: Learning and updating node embedding on dynamic heterogeneous information network. In: In Proc. of the 14th ACM International Conference on Web Search and Data Mining (WSDM), pp. 184–192. ACM (2021). DOI 10.1145/3437963.3441745
99. Xie, Y., Zhang, Y., Gong, M., Tang, Z., Han, C.: MGAT: multi-view graph attention networks. Neural Networks **132**, 180–189 (2020). DOI 10.1016/j.neunet.2020.08.021
100. Xiong, H., Yan, J., Pan, L.: Contrastive multi-view multiplex network embedding with applications to robust network alignment. In: In Proc. of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 1913–1923. ACM (2021). DOI 10.1145/3447548.3467227
101. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? CoRR **abs/1810.00826** (2018)
102. Xue, H., Yang, L., Jiang, W., Wei, Y., Hu, Y., Lin, Y.: Modeling dynamic heterogeneous network for link prediction using hierarchical attention with temporal RNN. CoRR **abs/2004.01024** (2020)
103. Yang, C., Xiao, Y., Zhang, Y., Sun, Y., Han, J.: Heterogeneous network representation learning: A unified framework with survey and benchmark. IEEE Trans. Knowl. Data Eng. **34**(10), 4854–4873 (2022). DOI 10.1109/TKDE.2020.3045924

104. Yang, L., Xiao, Z., Jiang, W., Wei, Y., Hu, Y., Wang, H.: Dynamic heterogeneous graph embedding using hierarchical attentions. In: In Proc. of the 42nd European Conference on Advances in Information Retrieval (ECIR), *Lecture Notes in Computer Science*, vol. 12036, pp. 425–432. Springer (2020). DOI 10.1007/978-3-030-45442-5_53
105. Yu, P., Fu, C., Yu, Y., Huang, C., Zhao, Z., Dong, J.: Multiplex heterogeneous graph convolutional network. In: In Proc. of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 2377–2387 (2022)
106. Zafarani, R., Abbasi, M.A., Liu, H.: Social media mining: an introduction. Cambridge University Press (2014)
107. Zangari, L., Interdonato, R., Caliò, A., Tagarelli, A.: Graph convolutional and attention models for entity classification in multilayer networks. Appl. Netw. Sci. **6**(1), 87 (2021). DOI 10.1007/s41109-021-00420-4
108. Zeng, H., Zhou, H., Srivastava, A., Kannan, R., Prasanna, V.: Graphsaint: Graph sampling based inductive learning method (2019). DOI 10.48550/ARXIV.1907.04931
109. Zhang, C., Song, D., Huang, C., Swami, A., Chawla, N.V.: Heterogeneous graph neural network. In: In Proc. of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 793–803. ACM (2019). DOI 10.1145/3292500.3330961
110. Zhao, J., Wang, X., Shi, C., Liu, Z., Ye, Y.: Network schema preserving heterogeneous information network embedding. In: In Proc. of the 29th International Joint Conference on Artificial Intelligence (IJCAI), pp. 1366–1372. ijcai.org (2020). DOI 10.24963/ijcai.2020/190
111. Zhou, F., Cao, C.: Overcoming catastrophic forgetting in graph neural networks with experience replay. In: In Proc. of the 35th AAAI Conference on Artificial Intelligence, 5, pp. 4714–4722. AAAI Press (2021)
112. Zhou, F., Xu, X., Li, C., Trajcevski, G., Zhong, T., Zhang, K.: A heterogeneous dynamical graph neural networks approach to quantify scientific impact. CoRR **abs/2003.12042** (2020)
113. Zhou, Y., Cheng, H., Yu, J.X.: Graph clustering based on structural/attribute similarities. Proc. VLDB Endow. **2**(1), 718–729 (2009). DOI 10.14778/1687627.1687709