Martirano *et al. Applied Network Science* 2022, **7**(1):65
https://doi.org/10.1007/s41109-022-00504-9

Applied Network Science

RESEARCH Open Access

Co-MLHAN: contrastive learning for multilayer heterogeneous attributed networks

Liliana Martirano, Lorenzo Zangari and Andrea Tagarelli*

*Correspondence:
tagarelli@dimes.unical.it

Department of Computer Engineering, Modeling, Electronics, and Systems Engineering (DIMES), University of Calabria, Rende, Italy

Abstract

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 complex networks. However, the existing methods consider only few aspects of networks at a time. In this paper, we propose a novel framework, named Co-MLHAN, to learn node embeddings for networks that are simultaneously *multilayer*, *heterogeneous* and *attributed*. 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.

Keywords: Graph representation learning, Contrastive learning, Multilayer networks, Heterogeneous networks, Attributed networks, Entity classification

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. 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).

Graph representation learning approaches are conventionally categorized into traditional embedding (a.k.a. "shallow") methods and *Graph Neural Network* (GNN)-based

methods. As noted in (Khoshraftar and An 2022), GNNs ensure 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 impact on scalability aspects.

Since GNNs typically require labels to learn rich representations, and 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 (Hassani and Ahmadi 2020). Among different graph self-supervised learning methods, contrast-based methods have more flexible designs and broader applications compared to other approaches (Liu et al. 2021), 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 (Jing et al. 2021).

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 Contrastive learning based framework for Multilayer Heterogeneous Attributed Networks (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 (Hassani and Ahmadi 2020; Jing et al. 2021; Mavromatis and Karypis 2021; Wang et al. 2021), 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 paper. The remainder of this paper is structured as follows. "Proposed framework" section describes our proposed framework in detail. "Experimental evaluation" section provides our experimental evaluation concerning the entity classification task on IMDb network datasets. "Related work" section discusses related works focusing on GNN-based approaches for representation learning in heterogeneous attributed networks and in multilayer attributed networks. "Conclusions" section contains concluding remarks and provides pointers for future research. Moreover, Appendices 1–4 provide details about the preprocessing of our evaluation network datasets, an insight into the content encoding stage, and a discussion on computational complexity aspects of the proposed framework.

Proposed framework

Our proposed Co-MLHAN is a self-supervised graph representation learning approach conceived for multilayer heterogeneous attributed networks. As previously discussed, a key novelty of Co-MLHAN 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 provide a formal definition of multilayer heterogeneous attributed graph and representation learning in such networks, then we move to a detailed description of Co-MLHAN. The notations used in this work are summarized in Table 9, Appendix 1.

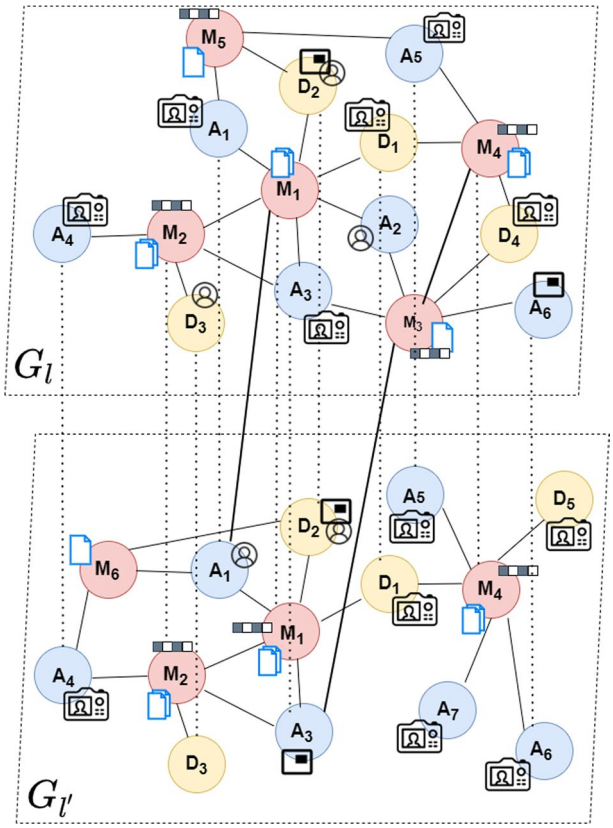

Fig. 1 Illustration of a multilayer heterogeneous attributed graph with two layers (G_l and $G_{l'}$), three types of nodes (M (MOVIE)–A (ACTOR)–D (DIRECTOR)), and different content features for different nodes (e.g., text, images, structured attributes)

Preliminary definitions

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 which nodes in one graph correspond to nodes in the other one. We assume that each layer can be *heterogeneous*, i.e., is characterized by nodes of different types 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 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*. Layers can be seen as different interaction contexts, semantic aspects, or time steps, while the participation of an entity to a layer can be seen as a particular *entity instance*. Instances of the same entity are connected via *pillar-edges*. We hereinafter refer to entity instances as *nodes* in the multilayer network. Figure 1 illustrates an example of a multilayer heterogeneous attributed graph.

Multilayer heterogeneous attributed graph. We define a *multilayer heterogeneous attributed graph* as $G_{\mathcal{L}} = \langle \mathcal{L}, \mathcal{V}, V_{\mathcal{L}}, E_{\mathcal{L}}, A, R, \phi, \varphi, \boldsymbol{\mathcal{X}}_{\mathcal{L}} \rangle$, 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, $\phi : \mathcal{V} \to A$ is the entity-type mapping function, $\varphi : E_{\mathcal{L}} \to R$ is the edge-type mapping function, and $\boldsymbol{\mathcal{X}}_{\mathcal{L}}$ is a set of matrices storing attributes, or initial features, with $\boldsymbol{\mathcal{X}}_{\mathcal{L}} = \bigcup_{l=1\dots\ell} \boldsymbol{\mathcal{X}}_l$. More specifically, entities, resp. nodes, of each type are assumed to be associated with features stored in layer-specific matrices $\boldsymbol{\mathcal{X}}_l = \{\mathbf{X}_l^{(a)}\}$, where each $\mathbf{X}_l^{(a)}$ is the feature 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 feature vector of entity v_i of type a in layer G_l. We also admit that features can be layer-independent, in which case we indicate with $\mathbf{x}_i^{(a)}$ the feature vector associated with entity v_i of type a in each layer, i.e., $\mathbf{x}_{\langle i,l\rangle}^{(a)} = \mathbf{x}_i^{(a)}$ for each $G_l \in \mathcal{L}$.

We specify that each entity has instances (i.e., nodes) in one or more layers, and appears at least in one layer, i.e., $\mathcal{V} = \bigcup_{l=1\dots\ell} \mathcal{V}_l$, with $\mathcal{V}_l$ set of entities appearing in the l-th layer. Likewise, $A = \bigcup_{l=1\dots\ell} A_l$, with A_l denoting the set of node types of the l-th layer, $R = \bigcup_{l=1\dots\ell} R_l$, 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, such as (Yang et al. 2021), in our formulation each layer G_l ($l = 1 \dots \ell$) is a heterogeneous graph at both node and edge levels, i.e., $|A_l| > 1$ *and* $|R_l| > 1$. Moreover, $A_l \subseteq A$, for all $G_l \in \mathcal{L}$, and $R_l \subset R$, since inter-layer connections are regarded as different types of edges.

Multilayer heterogeneous attributed graph embedding. Given a multilayer heterogeneous attributed network $G_{\mathcal{L}}$, 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 $v_i \in \mathcal{V}$, 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.

Co-MLHAN: contrastive learning framework for multilayer heterogeneous attributed networks

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

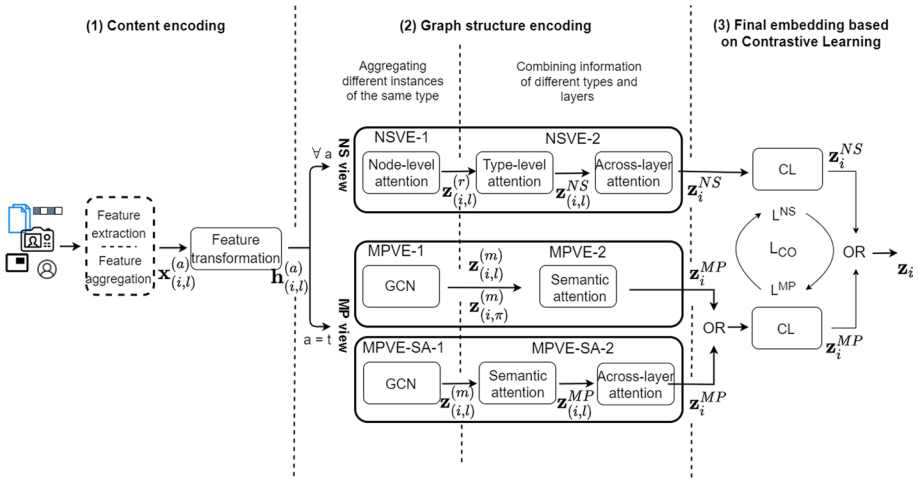

Fig. 2 Illustration of the three stages of the proposed Co-MLHAN framework

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 (Linsker 1988), 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 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 2 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

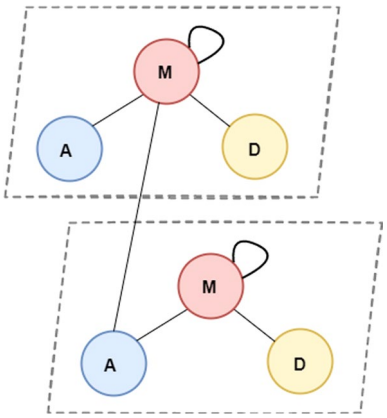

Fig. 3 Network schema graph corresponding to the multilayer heterogeneous attributed graph depicted in Fig. 1

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: (1) embeddings for the local structure ($\mathbf{z}^{\mathrm{NS}}$), including information from all direct neighbors of the nodes being instances of the target entity, and (2) embeddings for the high-order structure ($\mathbf{z}^{\mathrm{MP}}$), 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 the graph structure encoding (stage 2) and the generation of the final embedding based on contrastive learning (stage 3). We examine their computational complexity aspects in Appendix 4. For the sake of readability, note also that, since the first stage of content encoding is actually beyond the objectives of this work, we discuss it in Appendix 3.

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.

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. Formally, a network schema is a directed graph defined over node types A, with edges as relation types from R. 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

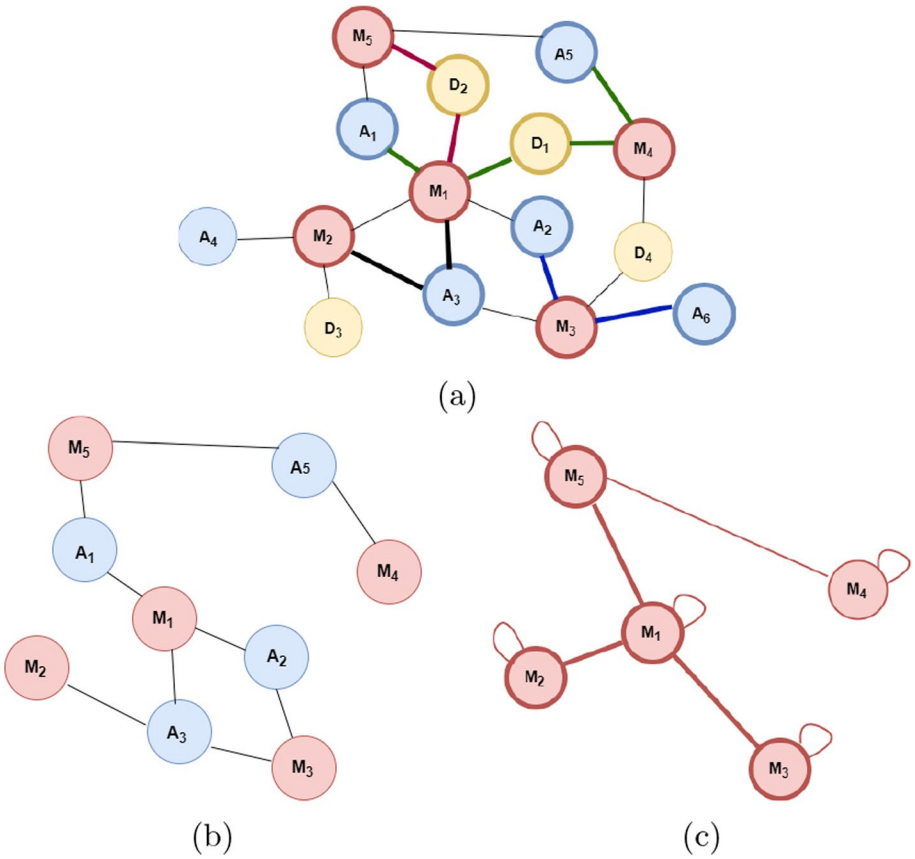

Fig. 4 Examples of meta-path instances for types MAM (MOVIE–ACTOR–MOVIE) marked with bold black lines, MDM (MOVIE–DIRECTOR–MOVIE) marked with bold red lines, AMA (ACTOR–MOVIE–ACTOR) marked with bold blue lines, and AMDMA (ACTOR–MOVIE–DIRECTOR–MOVIE–ACTOR) marked with bold green lines, w.r.t. layer G_l in the multilayer graph of Fig. 1 (**a**), all meta-path instances of type MAM w.r.t. the same layer (**b**) and the corresponding meta-path based graph for MAM type, with focus on node M_1 and its neighbors (**c**)

$R_{\langle i,l\rangle} \subseteq R$, and all node types a connected to the target node through a relation $r \in R_{\langle i,l\rangle}$. Hereinafter, we refer to this graph as **network schema graph**. Figure 3 shows an example of network schema graph for the multilayer heterogeneous attributed network of Fig. 1.

A **meta-path** is a sequence of connected nodes making two distant nodes in the network reachable, i.e., the *terminal* or *endpoint nodes* of a *meta-path instance*. Formally, a meta-path M_m is a path defined on the network schema graph, in the form $a_1 \xrightarrow{r_1} a_2 \xrightarrow{r_2} \cdots \xrightarrow{r_k} a_{k+1}$, describing a composite relation $r_1 \circ r_2 \circ \cdots \circ r_k$ between node types a_1 and a_{k+1}. A **meta-path instance** of M_m is a sampling under the guidance of M_m providing a sequence of connected nodes with edges matching the composite relation in M_m. Examples of within layer meta-path instances are depicted in Fig. 4a and b. Given a multilayer heterogeneous graph $G_{\mathcal{L}}$ and a meta-path M_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 the set of nodes of type a' that are connected with node $\langle i,l\rangle$ through at least one meta-path instance of M_m having a as starting node-type and a' as ending node-type. Note that, similarly to Wang et al. (2019), the intermediate nodes along meta-paths are discarded. A **meta-path based graph** is a graph comprised of all the meta-path based neighbors. For meta-paths

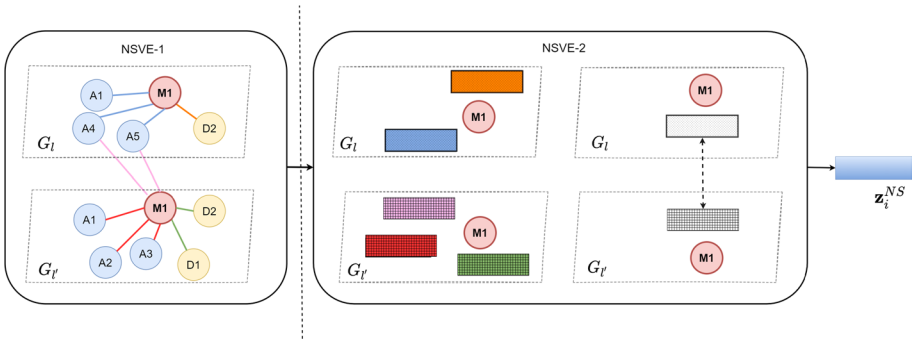

Fig. 5 Illustration of hierarchical attention approach used in the steps of the network schema view embedding (i.e., NSVE-1 and NSVE-2), with focus on the target entity M_1 in different layers. From left to right, NSVE-1 box shows node-level attention w.r.t. target nodes M_1, with colored lines denoting 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

with terminal nodes of the same type, the resulting graph is homogeneous at node level. Figure 4c shows an example of single-layer meta-path based graph according to a specific meta-path type.

Following Wang et al. (2021), given a target entity, the 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: (1) 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 (2) combining information of different types—i.e., different types of relations and of meta-paths, respectively, as well as different layers.

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 Fig. 5:

(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.

Given the graph $G_{\mathcal{L}}$, we define a function, denoted as $N^{(r)}(\cdot)$, 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 target node $\langle i, l\rangle$, we define the set of its neighbors under relation $r \in R_{\langle i,l\rangle}$ as:

$$N^{(r)}(i,l) = \{\langle j, l'\rangle \in V_{\mathcal{L}} | (\langle j, l'\rangle, \langle i, l\rangle) \in E_r\}. \tag{1}$$

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. (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. "Experimental settings" section). 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 v_i in layer G_l based on neighbors under relation r as:

$$\mathbf{z}_{\langle i,l\rangle}^{N^{(r)}} = \sigma\left(\sum_{\langle j,l'\rangle \in N^{(r)}(i,l)} \alpha_{\langle i,l\rangle,\langle j,l'\rangle}^{(r)} \mathbf{W}_{\mathbf{2}}^{(r)} \mathbf{h}_{\langle j,l'\rangle}\right), \tag{2}$$

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*), $\mathbf{W}_{\mathbf{2}}^{(r)}$ 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 $\alpha^{(r)}_{\langle i,l \rangle, \langle j,l' \rangle}$ 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:

$$\alpha^{(r)}_{\langle i,l \rangle, \langle j,l' \rangle} = \frac{\exp\left(e^{(r)}_{\langle i,l \rangle, \langle j,l' \rangle}\right)}{\sum_{\langle u,l' \rangle \in N^{(r)}(i,l)} \exp\left(e^{(r)}_{\langle i,l \rangle, \langle u,l' \rangle}\right)},$$
$$\text{with} \quad e^{(r)}_{\langle i,l \rangle, \langle j,l' \rangle} = \mathbf{a}^{(r)^{\mathrm{T}}}\left(LeakyReLU\left(\mathbf{W}^{(r)}\left[\mathbf{h}_{\langle i,l \rangle} \| \mathbf{h}_{\langle j,l' \rangle}\right]\right)\right), \tag{3}$$

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, $\mathbf{W}^{(r)} = [\mathbf{W}^{(r)}_{\mathbf{1}} \| \mathbf{W}^{(r)}_{\mathbf{2}}] \in \mathbb{R}^{d \times 2d}$ is the column-wise concatenation of $\mathbf{W}^{(r)}_{\mathbf{1}}$ and $\mathbf{W}^{(r)}_{\mathbf{2}}$, 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, we adopt the same approach as in GATv2 (Brody et al. 2021), which aims to fix the static attention problem of standard Graph Attention Network (GAT) (Velickovic et al. 2018) 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 Vaswani et al. (2017) 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:

$$\mathbf{z}^{N^{(r)}}_{\langle i,l \rangle} = \sigma\left(\bigoplus_{q=1\ldots Q}\left(\sum_{\langle j,l' \rangle \in N^{(r)}(i,l)} \alpha^{(r,q)}_{\langle i,l \rangle, \langle j,l' \rangle} \mathbf{W}^{(r,q)} \mathbf{h}_{\langle j,l' \rangle}\right)\right), \tag{4}$$

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

Let $\mathbf{z}^{N^{(r)}}_{\langle i,l \rangle}$ 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 $\bigcup_{r \in R_{\langle i,l \rangle}} \{\mathbf{z}^{N^{(r)}}_{\langle i,l \rangle}\}$ 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 $\mathbf{z}^{\mathrm{NS}}_{\langle i,l \rangle}$, as defined in Eq. 5:

[1] Alternatively, this operation can be carried out as $\mathbf{W}^{(r)}[\mathbf{h}_{\langle i,l \rangle} \| \mathbf{h}_{\langle j,l' \rangle}] = \mathbf{W}^{(r)}_{\mathbf{1}} \mathbf{h}_{\langle i,l \rangle} + \mathbf{W}^{(r)}_{\mathbf{2}} \mathbf{h}_{\langle j,l \rangle}$. Note that in order to save the number of parameters, $\mathbf{W}^{(r)}$ can be constrained to $[\mathbf{W}^{(r)}_{\mathbf{1}} \| \mathbf{W}^{(r)}_{\mathbf{1}}]$.

$$\mathbf{z}^{\mathrm{NS}}_{\langle i,l\rangle} = \sum_{r\in R_{\langle i,l\rangle}} \beta^{(r)} \mathbf{z}^{N^{(r)}}_{\langle i,l\rangle}, \tag{5}$$

where $\beta^{(r)}$ is the attention coefficient for neighborhood under relation r, which is defined as follows:

$$\beta^{(r)} = \frac{\exp\left(w^{(r)}\right)}{\sum_{r'\in R_{\langle i,l\rangle}} \exp\left(w^{(r')}\right)} \quad \text{with} \quad w^{(r)} = \frac{1}{|\mathcal{V}_l^{(t)}|} \sum_{\langle i,l\rangle\in\mathcal{V}_l^{(t)}} \mathbf{a}^{\mathrm{NS}^{\mathrm{T}}} \tanh\left(\mathbf{W}^{\mathrm{NS}} \mathbf{z}^{N^{(r)}}_{\langle i,l\rangle} + \mathbf{b}^{\mathrm{NS}}\right), \tag{6}$$

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 $\bigcup_{l\in L}\{\mathbf{z}^{\mathrm{NS}}_{\langle i,l\rangle}\}$ 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 v_i, as defined in Eq. 7:

$$\mathbf{z}^{\mathrm{NS}}_i = \sum_{l\in L} \beta^{(l)} \mathbf{z}^{\mathrm{NS}}_{\langle i,l\rangle}, \tag{7}$$

where $\beta^{(l)}$ is the learned attention coefficient for layer G_l, computed via the same attention model like in Eq. 6, where in this case the learnable weights are shared by all layers.

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 (Lin et al. 2021), 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 (Fig. 6):

(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) (Kipf and Welling 2017), obtaining, for each target node, an embedding w.r.t. each meta-path type.

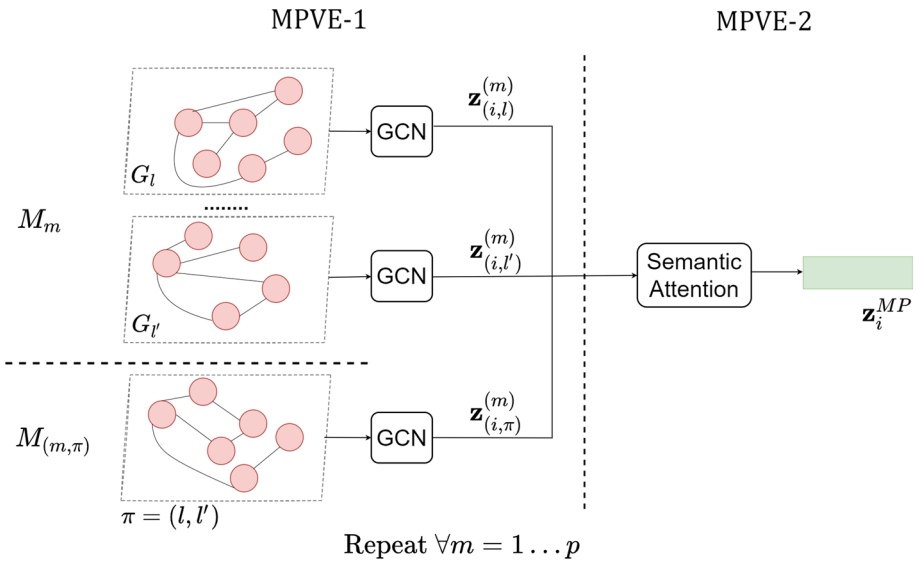

Fig. 6 Illustration of the sub-steps of the Co-MLHAN embedding generation under the meta-path view of the entity v_i

(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 "Alternative meta-path view embedding: Co-MLHAN-SA" section, 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 $\mathcal{M} = \{M_1, \ldots, M_p\}$ involving nodes of target type, where each meta-path M_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*. They refer to the same composite relation, 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 (of type different from the target one) with both instances involved in the composite relation. An example is illustrated in Fig. 7. We define the set of *across-layer meta-paths*, $\mathcal{M}^{\updownarrow}$, 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:

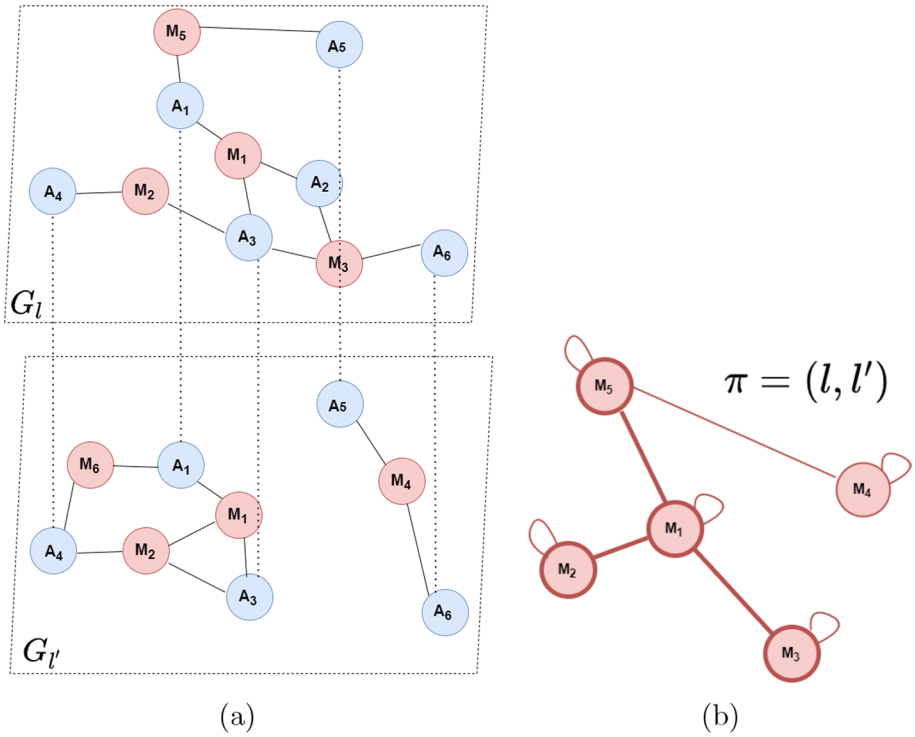

Fig. 7 All across-layer meta-path instances of type MAM (**a**), and the corresponding meta-path based graph for MAM type, with focus on entity *M*1 and its neighbors (**b**)

$$N_m^{\Leftrightarrow}(i,l) = \{\langle j,l\rangle \in V_{\mathcal{L}} \mid \langle j,l\rangle \in N_m(i,l)\}. \tag{8}$$

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

$$N_m^{\Updownarrow}(i,l) = \{\langle j,l'\rangle \in V_{\mathcal{L}} \mid \langle j,l'\rangle \in N_m(i,l), l' \neq l\}. \tag{9}$$

Note that Eqs. 8 and 9 identify the meta-path based neighborhood of type M_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:

$$\mathbf{z}_{\langle i,l\rangle}^{(k+1)} = \begin{cases} f_m^{(k+1)}\left(\mathbf{z}_{\langle i,l\rangle}^{(k)}, \bigoplus\left\{\mathbf{z}_{\langle j,l\rangle}^{(k)} \mid \langle j,l\rangle \in N_m^{\Leftrightarrow}(i,l)\right\}\right) & \text{if } m \text{ is a within layer meta-path} \\ f_m^{(k+1)}\left(\mathbf{z}_{\langle i,l\rangle}^{(k)}, \bigoplus\left\{\mathbf{z}_{\langle j,l\rangle}^{(k)} \mid \langle j,l\rangle \in N_m^{\Updownarrow}(i,l)\right\}\right) & \text{if } m \text{ is an across-layer meta-path} \end{cases} \tag{10}$$

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

(Wang et al. 2020)]. Similarly to Wang et al. (2021), we use a GCN architecture as f_m, for all M_m ($m = 1 \dots p$) in Eq. 10, assuming no different contribution from different instances of the same meta-path.

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

$$\mathbf{z}_{\langle i,l\rangle}^{(k+1)} = \sigma\left(\sum_{\langle j,l\rangle \in N^{\Leftrightarrow}(i,l)} \frac{1}{\sqrt{\widetilde{\mathbf{D}}_{ii}^{l}\widetilde{\mathbf{D}}_{jj}^{l}}} \mathbf{W}^{(k,l)^{\mathrm{T}}} \mathbf{z}_{\langle j,l\rangle}^{(k)}\right) \tag{11}$$

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 $\widetilde{\mathbf{D}}_{ii}^{l} = \sum_j \widetilde{\mathbf{A}}_{ij}^{l}$ is the degree matrix derived from $\widetilde{\mathbf{A}}_l = \mathbf{A}_l + \mathbf{I}_n$, with $\mathbf{I}_n^l$ 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 $\mathbf{z}_{\langle i,l\rangle}^{(m)}$ and $\mathbf{z}_{\langle i,\pi\rangle}^{(m)}$ 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 $\{\mathbf{z}_{\langle i,l\rangle}^{(m)} \mid l \in L,\ m = 1 \dots p\} \bigcup \{\mathbf{z}_{\langle i,\pi\rangle}^{(m)} \mid \langle m, \pi\rangle \in \mathcal{M}^{\Updownarrow}\}$ 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:

$$\mathbf{z}_{\langle i,l\rangle}^{\mathrm{MP}} = \sum_{m=1}^{p} \beta^{(m,l)} \mathbf{z}_{\langle i,l\rangle}^{(m)} + \lambda^{\Updownarrow}\left(\sum_{m=1}^{p} \sum_{\pi \mid l \in \pi} \beta^{(m,\pi)} \mathbf{z}_{\langle i,\pi\rangle}^{(m)}\right), \tag{12}$$

where β is the attention coefficient denoting the importance of each type of within layer and across-layers meta-path (cf. Eq. 6) and $\lambda^{\Updownarrow} \in [0 \dots 1]$ 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:

$$\mathbf{z}_i^{\mathrm{MP}} = \sum_{l \in L} \mathbf{z}_{\langle i,l\rangle}^{\mathrm{MP}}. \tag{13}$$

Note that Eq. 13 does not require an additional level of attention, since the layer dependency has already been taken into account by the attention mechanism in Eq. 12. Therefore, Eqs. 12 and 13 can be combined as follows:

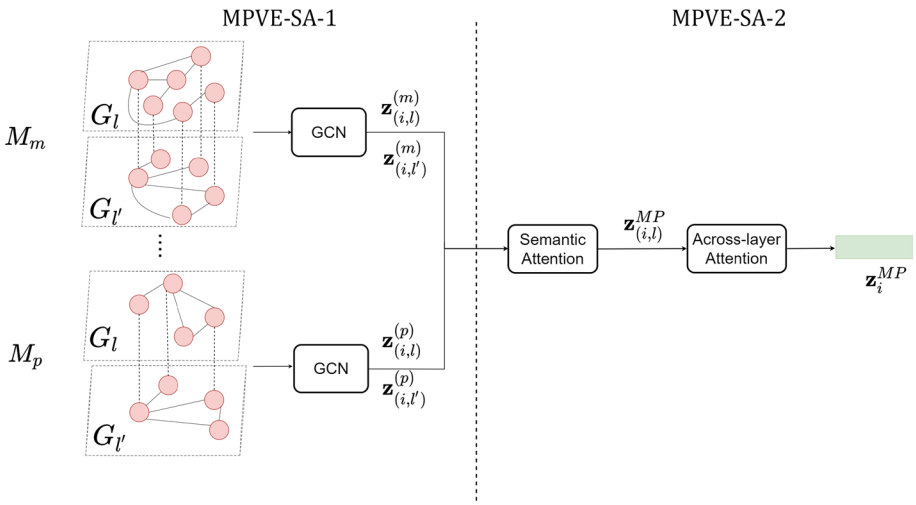

Fig. 8 Illustration of the sub-steps of the Co-MLHAN-SA embedding generation under the meta-path view of the entity v_i

$$\mathbf{z}_i^{\mathrm{MP}} = \underbrace{\sum_{m=1}^{p} \sum_{l \in L} \beta^{(m,l)} \mathbf{z}^{(m)}_{\langle i,l \rangle}}_{\text{within-layer}} + \underbrace{\lambda^{\updownarrow} \left(\sum_{m=1}^{p} \sum_{\pi \in L_{cross}} \beta^{(m,\pi)} \mathbf{z}^{(m)}_{\langle i,\pi \rangle} \right)}_{\text{across-layers}}. \tag{14}$$

Equation 14 hence enables the direct computation of the final embedding under the meta-path view for each entity v_i.

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 $\mathbf{A}^{\mathrm{sup}}$, has diagonal blocks each representing a layer-specific adjacency matrix (i.e., $\mathbf{A}_l \in \mathbb{R}^{n_l \times n_l}$, with $l = 1 \ldots \ell$), 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 (Fig. 8):

(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 $N^{\leftrightarrows}(\cdot)$ and $N^{\updownarrow}(\cdot)$ to indicate the set of within-layer and across-layer neighbors, respectively. While the definition of $N^{\leftrightarrows}(\cdot)$ does not change w.r.t. Eq. 8, 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, as shown in Eq. 15:

$$N_m^{\updownarrow}(i,l) = \{\langle i,l'\rangle \in V_{\mathcal{L}} \mid l' \neq l\}. \tag{15}$$

Similarly to MPVE-1, we apply a meta-path specific GNN for aggregating different meta-path instances of the same type:

$$\mathbf{z}_{\langle i,l\rangle}^{(k+1)} = f_m^{(k+1)}\left(\mathbf{z}_{\langle i,l\rangle}^{(k)}, \bigoplus^{k}\left(\left\{\mathbf{h}_{\langle j,l\rangle}^{(k)} \mid \langle j,l\rangle \in N^{\leftrightarrows}(i,l) \cup N^{\updownarrow}(i,l)\right\}\right)\right). \tag{16}$$

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 (Zangari et al. 2021), as shown in Eq. 17:

$$\mathbf{z}_{\langle i,l\rangle}^{(k+1)} = \sigma\left(\sum_{\langle j,l'\rangle \in N^{\leftrightarrows}(i,l)\cup N^{\updownarrow}(i,l)} \frac{1}{\sqrt{\widetilde{\mathbf{D}}_{ii}\widetilde{\mathbf{D}}_{jj}}} \mathbf{W}^{(k,m)^{\mathrm{T}}} \delta(l,l')\, \mathbf{z}_{\langle j,l'\rangle}^{(k)}\right), \tag{17}$$

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, $\widetilde{\mathbf{D}}_{ii} = \sum_{j=1} \widetilde{\mathbf{A}}_{ij}^{\text{sup}}$, 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 $\mathbf{z}_{\langle i,l\rangle}^{m}$ be the embedding of node $\langle i,l\rangle$ associated with the m-th metapath. We thus obtain $\bigcup_{\substack{m=1\ldots p \\ l\in L}} \{\mathbf{z}_{\langle i,l\rangle}^{(m)}\}$ 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:

$$\mathbf{z}^{\text{MP}}_{\langle i,l \rangle} = \sum_{m=1 \ldots p} \beta^{(m,l)} \mathbf{z}^{(m)}_{\langle i,l \rangle}, \tag{18}$$

where $\beta^{(m,l)}$ is an attention coefficient computed as in Eq. 6.

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 v_i an embedding under the meta-path view:

$$\mathbf{z}^{\text{MP}}_{i} = \sum_{l \in L} \beta^{(l)} \mathbf{z}^{\text{MP}}_{\langle i,l \rangle}, \tag{19}$$

where $\beta^{(l)}$ is the attention coefficient denoting the importance of the l-th layer, computed similarly to Eq. 6.

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. 7) and $\mathbf{z}^{\text{MP}}_i$ (either Eq. 14 or Eq. 19) for each target entity v_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. 20:

$$\begin{aligned} \hat{\mathbf{z}}^{\text{NS}}_i &= \mathbf{W}^{(2)} \sigma(\mathbf{W}^{(1)} \mathbf{z}^{\text{NS}}_i + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}, \\ \hat{\mathbf{z}}^{\text{MP}}_i &= \mathbf{W}^{(2)} \sigma(\mathbf{W}^{(1)} \mathbf{z}^{\text{MP}}_i + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}, \end{aligned} \tag{20}$$

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 (Khosla et al. 2020; Wang et al. 2021). Each target entity v_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 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. 21:

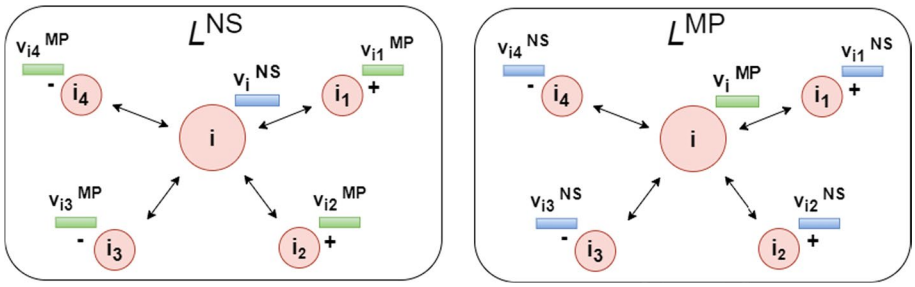

Fig. 9 Illustration of multi-view contrastive learning. For L^{NS}, the embedding of the target entity v_i is under the network schema view (colored in blue), while positive (+) and negative (−) samples are under the meta-path view (colored in green). By contrast, for L^{MP}, the embedding of the target entity v_i is under the meta-path view (colored in green), while positive (+) and negative (−) samples are under the network schema view (colored in blue)

$$C_{i,j} = \sum_{l \in L} \sum_{m=1 \ldots p} |\{j \mid \langle j, l \rangle \in N_m^{\Leftrightarrow}(i,l)\}|. \tag{21}$$

For each target entity v_i, we obtain a set $\mathcal{S}_i = \{v_j \in \mathcal{V} \mid C_{i,j} > 0\}$ 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 v_i. Therefore, for each entity v_i, we define the set of **positive samples** $\mathcal{P}_i$ as $\mathcal{P}_i = v_i \cup \{v_j | v_j \in \overline{\mathcal{S}}_i\}$ and the set of **negative samples** $\mathcal{N}_i$ as $\mathcal{N}_i = \mathcal{V} \backslash \mathcal{P}_i$.

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 v_i is selected from the given view, while the positive and negative samples are selected from the other view, as defined in Eqs. 22 and 23, and illustrated in Fig. 9:

$$L^{\mathrm{NS}} = -\log \frac{\sum_{j \in \mathcal{P}_i} \exp\left(sim\left(\hat{\mathbf{z}}_i^{\mathrm{NS}}, \hat{\mathbf{z}}_j^{\mathrm{MP}}\right)/\tau\right)}{\sum_{u \in \mathcal{P}_i \cup \mathcal{N}_i} \exp\left(sim\left(\hat{\mathbf{z}}_i^{\mathrm{NS}}, \hat{\mathbf{z}}_u^{\mathrm{MP}}\right)/\tau\right)}, \tag{22}$$

$$L^{\mathrm{MP}} = -\log \frac{\sum_{j \in \mathcal{P}_i} \exp\left(sim\left(\hat{\mathbf{z}}_i^{\mathrm{MP}}, \hat{\mathbf{z}}_j^{\mathrm{NS}}\right)/\tau\right)}{\sum_{u \in \mathcal{P}_i \cup \mathcal{N}_i} \exp\left(sim\left(\hat{\mathbf{z}}_i^{\mathrm{MP}}, \hat{\mathbf{z}}_u^{\mathrm{NS}}\right)/\tau\right)}, \tag{23}$$

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. 22–23 are independent from the specific strategy of positive and negative selection; we leave the investigation of alternative sampling methods as future work ("Conclusions" section).

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

$$L_{co} = \lambda L^{\mathrm{NS}} + (1 - \lambda) L^{\mathrm{MP}} \tag{24}$$

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. 25:

$$L_{tot} = \eta L_{co} + L_{sup} \tag{25}$$

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 Chen et al. (2020), 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 v_i includes itself under the other view in its set of positive samples $\mathcal{P}_i$. Nonetheless, in "Experimental settings" section, 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.

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. "Data" section introduces the data, "Competing methods" section presents the competing methods, "Experimental settings" section discusses the experimental settings, and "Results" section describes the main results.

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

Table 1 Summary of within-layer network statistics

	IMDb-MLH		*IMDb-MLH-mb*	
	2020	**2021**	**2020**	**2021**
Nodes				
MOVIE	1852	1459	1992	1580
ACTOR	9165	7364	10,044	8096
DIRECTOR	3937	3003	3964	3019
Edges				
MOVIE–ACTOR	10,966	8741	12,203	9751
MOVIE–DIRECTOR	4972	3838	5002	3860
Meta-paths				
MOVIE–ACTOR–MOVIE	3862	3096	5346	4414
MOVIE–DIRECTOR–MOVIE	1874	1310	1884	1328

Table 2 Summary of across-layer network statistics

	IMDb-MLH	*IMDb-MLH-mb*
Entities		
MOVIE	2807	3033
ACTOR	14,720	15,987
DIRECTOR	5736	5775
Pillar edges		
MOVIE	504	539
ACTOR	1809	2153
DIRECTOR	1204	1208
Meta-paths		
MOVIE–ACTOR–MOVIE	3417	4629
MOVIE–DIRECTOR–MOVIE	1697	1701

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., Wang et al. 2019; Fu et al. 2020; Zhao et al. 2020) 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 DIRECTOR (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',

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

Table 3 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%

'comedy' and 'drama'. Tables 1, 2 and 3 summarize main characteristics of the networks, which are described next, whereas in Appendix 2, we provide a detailed description of the semantics of the constituting elements and the steps involved for data preprocessing.

- *IMDb-MLH.* Our main network dataset was conceived primarily for comparative evaluation with the competitors. As it can be noticed from Table 3, 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 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.

Competing methods

We compared Co-MLHAN and Co-MLHAN-SA with two unsupervised learning methods, HeCo (Wang et al. 2021) and NSHE (Zhao et al. 2020), 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 methods, NSHE generates initial features of nodes by using DeepWalk (Perozzi et al. 2014) 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. "Related work" section).

Table 4 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
T_{pos}	5	5	5

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 both 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.

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 4.

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 strategy and the joint optimization of the loss function entail similar representations. To validate our hypothesis, for each entity v_i, we computed the cosine similarity between the embedding under the network schema view ($\mathbf{z}_i^{\mathrm{NS}}$) and the embedding under the meta-path view ($\mathbf{z}_i^{\mathrm{MP}}$). 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 (Kingma and Ba 2017) with full batch size, for 10,000 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 **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 (Li et al. 2019)]. 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 Wang et al. (2021), 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., (Zangari et al. 2021).

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. "Competing methods" section), for NSHE we selected the same hyperparameters it uses for the IMDb dataset (Zhao et al. 2020). 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, 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

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

Table 5 Results on *IMDb-MLH* (1000 features), with and without early-stopping

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

Bold values refer to the best score for each criterion

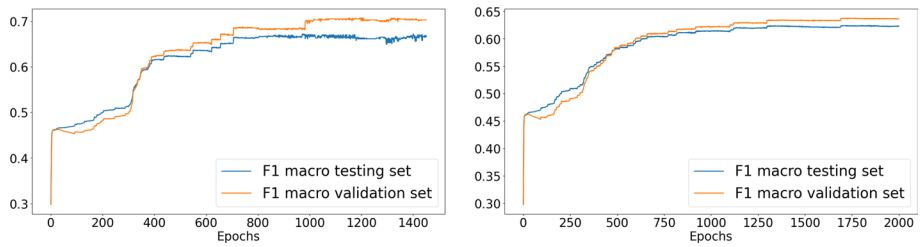

Fig. 10 Testing and validation F1 macro for the final classifier with early-stopping (left) and without early-stopping (right)

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.

Results

We organize the presentation of our experimental results into four parts: "Evaluation on *IMDb-MLH*" and "Evaluation on *IMDb-MLH-mb*" sections concern the evaluation on *IMDb-MLH* and *IMDb-MLH-mb*, respectively, whereas "Qualitative inspection of the embeddings" section provides a qualitative analysis of the learned embeddings. Finally, "Summary of results" section summarizes our experimental findings.

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 2); 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 5, 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 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 2, only 504 out of 2807 target entities are shared between the two layers.

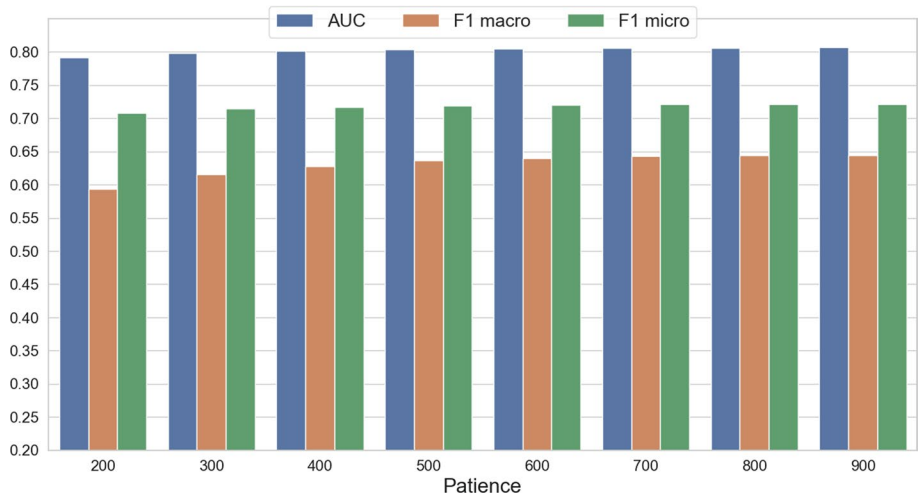

Fig. 11 Quality criteria of the final classifier with different patience values. The best performance is reached with patience equal to 900: 0.807 AUC, 0.645 F1 macro, 0.721 F1 micro

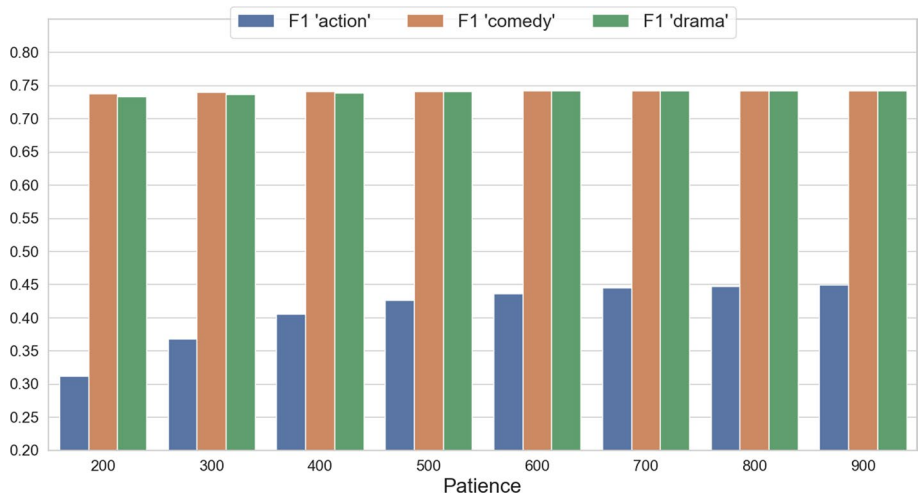

Fig. 12 Quality criteria on each class of the final classifier with early-stopping technique with different patience values

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.

Table 6 Results on *IMDb-MLH* (4000 features), with early stopping

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

Bold values refer to the best score for each criterion

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 5, 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, Fig. 10 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. 10, 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 around the 1000-th epoch (Fig. 10, right); however, as shown in Fig. 11, 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 Fig. 12.

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 setting

Table 7 Results on *IMDb-MLH-mb* (1000 features) and positives selection *AL3A*

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

Bold values refer to the best score for each criterion

Table 8 Results on *IMDb-MLH-mb* (1000 features) and positives selection *AL1A*

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**

Bold values refer to the best score for each criterion

are reported in Table 6 (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 (Zhao et al. 2020). 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.

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

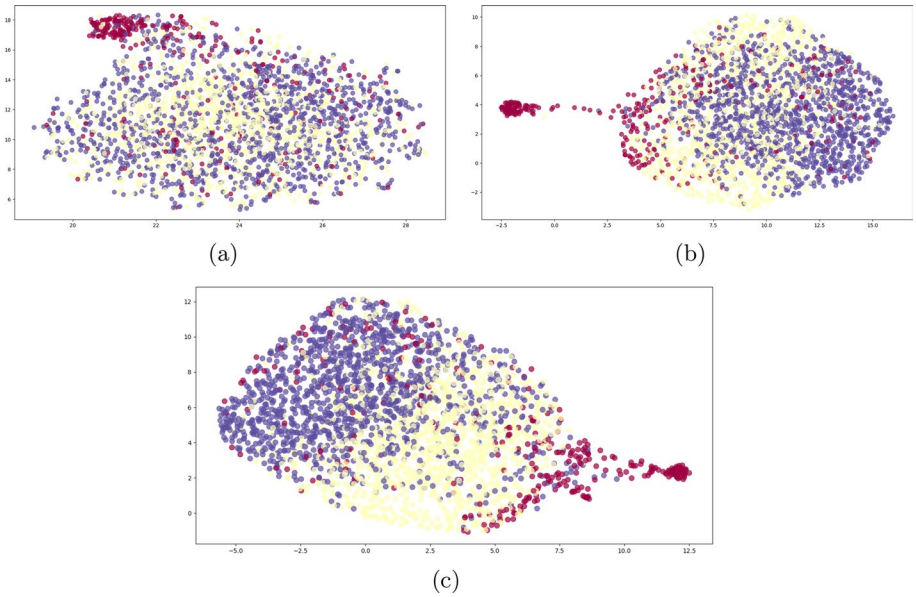

Fig. 13 UMAP 2D visualization of the entity embeddings on the *IMDb-MLH* dataset, with red, yellow and purple points indicating movies of genre 'action', 'comedy' and 'drama', respectively: the initial feature embedding with tf-idf weighting (**a**), the final entity embedding learned by Co-MLHAN-SA (**b**), and the final entity embedding learned by Co-MLHAN (**c**)

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 7, 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 4 and Appendix 2 for additional details) to test the sensitivity of our methods, without changing the feature initialization. Results shown in Table 8 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 2); nonetheless, we observed a negligible worsening in the performance.

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 and Projection (UMAP) (McInnes et al. 2018), 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 (van der Maaten and Hinton 2008). 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 13 shows the two-dimensional UMAP visualization of the initial feature embeddings with tf-idf weighting (Fig. 13a), and of the final embeddings under the meta-path view learned by our methods (Fig. 13b, c), 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. 13a), 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, Fig. 13b and c show how the final embeddings learned by our methods allow UMAP to better separate entities of different classes.

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 2), 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.

Related work

We discuss below most relevant GNN-based approaches that are designed for different aspects of complex networks and particularly related to our approach. Over the last years, several works focused on the extension of popular GNN models such as GCN (Kipf and Welling 2017) and GAT (Velickovic et al. 2018) to the heterogeneous or multilayer case. Their extension is still an open research problem. In this section, we explore both semi-supervised and unsupervised learning paradigms, with emphasis on contrastive learning approaches in unsupervised contexts.

Representation learning for heterogeneous attributed networks

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 (Zhang et al. 2019) 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 (Wang et al. 2019) 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 (Fu et al. 2020) 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 (Manchanda et al. 2021) 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 (Hu et al. 2020) 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. Co-MLHAN supports a user-specified selection of meta-paths and focuses on meta-path based neighbors of target type. We discard the information of intermediate nodes, according to the idea of differentiating local and high-order information in distinct views.

More recently developed approaches rely on considering node local and high-order structure separately. NSHE (Zhao et al. 2020) 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 (Wang et al. 2021) 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 (Khan and Kleinsteuber 2021) 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, Co-MLHAN adopts 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.

Representation learning for multilayer networks

Some major challenges for multilayer 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 (Grassia et al. 2021) 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 (Ghorbani et al. 2019) 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 (Ma et al. 2019), which models explicit adjacency links among different layers. mGAT (Xie et al. 2020) is an attention-based approach that introduces a regularization term to the loss function to constrain the similarity between each pair of layers. GrAMME (Shanthamallu et al. 2020) 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 (Zangari et al. 2021) 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. Co-MLHAN 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 (Liu et al. 2021) 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 (Shi et al. 2021) 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 Co-MLHAN, the two views in AMC-GNN are obtained exploiting data augmentation on the original graph. DMGI (Park et al. 2019) 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 (Xiong et al. 2021) 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. Co-MLHAN 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.

Representation learning for heterogeneous attributed multilayer networks

In the past few years, interest has started to emerge in combining heterogeneity and multilayer aspects, however literature still lacks works focusing on embedding generation for such networks. GATNE (Cen et al. 2019) 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. Co-MLHAN also employs meta-paths to capture high-order relations between nodes, although the meta-path types are specified at the modeling stage. Moreover, we learn a single encompassing embedding for each node/entity, incorporating different relation types.

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 proposed framework.

Conclusions

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 enables handling across-layer information, which we handle by two versions of the framework differing in the modeling of pillar edges: Co-MLHAN, modeling a particular type of meta-paths with terminal nodes belonging to different layers and the intermediate node—of a different type from target—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 handle 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 we discuss in Appendix 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 (Hamilton et al. 2018; Chen et al. 2018; Zeng et al. 2019; Hu et al. 2020).

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 (Hassani and Ahmadi 2020)), 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 (Kalantidis et al. 2020; Ahrabian et al. 2020; Robinson et al. 2020).

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 (Li et al. 2020).

Appendix

Appendix 1. Notations

Table 9 summarizes main notations used throughout this work.

Table 9 Summary of notations and their description

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
$\boldsymbol{\mathcal{X}}_{\mathcal{L}}, \boldsymbol{\mathcal{X}}_l, \mathbf{X}_l^{(a)}$	Sets of attribute matrices and layer-specific matrices, and set of attribute matrix 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
$\mathbf{z}_i^{\mathrm{NS}}, \mathbf{z}_{\langle i,l\rangle}^{\mathrm{NS}}$	Embedding of entity i and node $\langle i,l\rangle$ under *network schema view*
$\mathcal{M}, M_m, p$	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
$\mathbf{z}_i^{\mathrm{MP}}, \mathbf{z}_{\langle i,l\rangle}^{\mathrm{MP}}$	Embedding of entity i and node $\langle i,l\rangle$ under *meta-path view*
$\hat{\mathbf{z}}_i^{\mathrm{NS}}, \hat{\mathbf{z}}_i^{\mathrm{MP}}$	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)
$T_{pos}, \overline{\mathcal{S}}_i$	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
$L^{\mathrm{NS}}, L^{\mathrm{MP}}, L_{co}, L_{sup}, L_{tot}$	Loss functions

Appendix 2. Data

In the following we provide details of our datasets built upon IMDb. 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: MAM (MOVIE–ACTOR–MOVIE) and MDM (MOVIE–DIRECTOR–MOVIE) for type M, from which we derive pairs of MOVIES starring the same ACTOR or directed by the same DIRECTOR, respectively; AMA (ACTOR–MOVIE–ACTOR) and AMDMA (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; DMD (DIRECTOR–MOVIE–DIRECTOR) and DMAMD (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 MAM and MDM, 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 MDM or MAM instance connecting the two entities, requiring at

least one MDM or one MAM, 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 MDM or MAM instance connecting the two entities, requiring at least one MDM or three MAMs, 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 4.

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 Fig. 7, given a meta-path of type MAM (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 (Kipf and Welling 2017). 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.

Appendix 3. 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 previously introduced, given a type $a \in A$, we denote with $\mathbf{x}_{\langle i,l \rangle}^{(a)}$ the initial feature vector of node $\langle i, l \rangle$ (i.e., entity v_i in layer G_l), and with $\mathbf{x}_i^{(a)}$ the initial feature vector of entity v_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 $\mathbf{h}_i^{(a)}$, for entity v_i of type a, as follows:

$$\mathbf{h}_i^{(a)} = \sigma(\mathbf{W}^{(a)}\mathbf{x}_i^{(a)} + \mathbf{b}^{(a)}), \tag{26}$$

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

$$\mathbf{h}_{\langle i,l\rangle}^{(a)} = \sigma(\mathbf{W}_l^{(a)}\mathbf{x}_{\langle i,l\rangle}^{(a)} + \mathbf{b}_l^{(a)}), \tag{27}$$

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. 26 and 27, $\sigma(\cdot)$ 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 (Baevski et al. 2022) which proposes a general self-supervised learning framework for generating contextualized latent representation of different modalities, including speech, images and text.

Appendix 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 $\mathcal{O}(|\mathcal{V}|\ell)$ and each target node appears in the meta-path based graphs, i.e., the total number of nodes in the meta-path view is $\mathcal{O}(|\mathcal{V}^{(t)}|\ell)$ 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 with a single attention head is $T_r = \mathcal{O}(|V_r|d^2 + |E_r|d)$ (Brody et al. 2021), 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 $\mathcal{O}(max(T_{r_1}, T_{r_2}, \ldots, T_{r_{|R|}}))$.

(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 $\mathcal{O}(|\mathcal{V}^{(t)}|d^3|R|)$, 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 $\mathcal{O}(|\mathcal{V}^{(t)}|\ell d^3)$. 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 (Kipf and Welling 2017), whose cost for K neural layers is $\mathcal{O}(Knonzero(\mathbf{A}_l)d + K|\mathcal{V}^{(t)}|d^2)$, 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 (Li et al. 2019), 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 $\mathcal{O}(p|\mathcal{V}^{(t)}|\ell d^3)$, 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 (Zangari et al. 2021) with K neural layers. Its computational complexity is $\mathcal{O}(Knonzero(\mathbf{A}^{\text{sup}})d + K|\mathcal{V}^{(t)}|\ell d^2)$, 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., $\mathcal{O}(p|\mathcal{V}^{(t)}|\ell d^3)$. 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., $\mathcal{O}(|\mathcal{V}^{(t)}|\ell d^3)$.

The third stage, based on contrastive learning, requires first a transformation through a MLP, which costs $\mathcal{O}(|\mathcal{V}^{(t)}|d^2)$, 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 $\mathcal{O}(|\mathcal{V}^{(t)}|^2 d)$.

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 (Hamilton et al. 2018), or investigating more efficient similarity methods.

Author contributions
LM and AT conceived the idea presented in this work. LM, LZ, and AT developed the theoretical definition of the methods. LM, LZ, and AT defined the evaluation methodology and experiments to perform. LM and LZ developed the code and took care of running the experiments. All authors performed evaluation of the results and related discussion. AT supervised the writing, reviewing and editing. All authors participated in the writing process. All authors read and approved the final manuscript.

Funding
LM was funded by the PON FSE-FESR Ricerca e Innovazione 2014–2020 (PON R &I), Azione I.1 "Dottorati Innovativi con caratterizzazione industriale", Avviso n. 1233, July 30, 2020. The paper was partially funded by POR CALABRIA FESR 2014/2020 "Smart Cities Lab" (CUP J89J21018490005, former J89J21009750007).

Availability of data and materials
Python code for the proposed methods, as well as the network datasets, are available at https://people.dimes.unical.it/andreatagarelli/co-mlhan/.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 31 May 2022 Accepted: 9 August 2022
Published: 20 September 2022

References

Ahrabian K, Feizi A, Salehi Y, Hamilton WL, Bose AJ (2020) Structure aware negative sampling in knowledge graphs. CoRR. arXiv:2009.11355
Baevski A, Hsu W-N, Xu Q, Babu A, Gu J, Auli M (2022) data2vec: a general framework for self-supervised learning in speech. Vis Lang arXiv. https://doi.org/10.48550/ARXIV.2202.03555
Brody S, Alon U, Yahav E (2021) How attentive are graph attention networks? CoRR. arXiv:2105.14491
Cen Y, Zou X, Zhang J, Yang H, Zhou J, Tang J (2019) Representation learning for attributed multiplex heterogeneous network. CoRR. arXiv:1905.01669
Chen J, Ma T, Xiao C (2018) Fastgcn: Fast learning with graph convolutional networks via importance sampling
Chen T, Kornblith S, Norouzi M, Hinton G (2020) A simple framework for contrastive learning of visual representations. https://doi.org/10.48550/ARXIV.2002.05709
Fu X, Zhang J, Meng Z, King I (2020) MAGNN: metapath aggregated graph neural network for heterogeneous graph embedding. CoRR. arXiv:2002.01680
Ghorbani M, Baghshah MS, Rabiee HR (2019) MGCN: semi-supervised classification in multi-layer graphs with graph convolutional networks. In: Spezzano F, Chen W, Xiao X (eds) ASONAM '19: international conference on advances in social networks analysis and mining, Vancouver, British Columbia, Canada, 27–30 August, 2019. ACM, pp 208–211. https://doi.org/10.1145/3341161.3342942
Grassia M, Domenico MD, Mangioni G (2021) mGNN: generalizing the graph neural networks to the multilayer case. CoRR. arXiv:2109.10119
Hamilton WL, Ying R, Leskovec J (2018) Inductive representation learning on large graphs. CoRR. arXiv:1706.02216
Hassani K, Ahmadi AHK (2020) Contrastive multi-view representation learning on graphs. CoRR. arXiv:2006.05582
Hu Z, Dong Y, Wang K, Sun Y (2020) Heterogeneous graph transformer. CoRR. arXiv:2003.01332
Jing B, Xiang Y, Chen X, Chen Y, Tong H (2021) Graph-mvp: multi-view prototypical contrastive learning for multiplex graphs. CoRR. arXiv:2109.03560
Kalantidis Y, Sariyildiz MB, Pion N, Weinzaepfel P, Larlus D (2020) Hard negative mixing for contrastive learning. CoRR. arXiv:2010.01028
Khan RA, Kleinsteuber M (2021) A framework for joint unsupervised learning of cluster-aware embedding for heterogeneous networks. CoRR. arXiv:2108.03953
Khoshraftar S, An A (2022) A survey on graph representation learning methods. CoRR. https://doi.org/10.48550/arXiv.2204.01855
Khosla P, Teterwak P, Wang C, Sarna A, Tian Y, Isola P, Maschinot A, Liu C, Krishnan D (2020) Supervised contrastive learning. CoRR. arXiv:2004.11362
Kingma DP, Ba J (2017) Adam: a method for stochastic optimization. CoRR arXiv:1412.6980

Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: Proceedings of the 5th international conference on learning representations (ICLR)
Li G, Muller M, Thabet A, Ghanem B (2019) Deepgcns: Can GCNS go as deep as CNNS? In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV)
Li H, Wang Y, Lyu Z, Shi J (2020) Multi-task learning for recommendation over heterogeneous information network. IEEE Trans Knowl Data Eng 34:789–802
Lin B, Wang X, Dong Y, Huo C, Ren W, Xu C (2021) Metapaths guided neighbors aggregated network for? Heterogeneous graph reasoning. https://doi.org/10.48550/ARXIV.2103.06474
Linsker R (1988) Self-organization in a perceptual network. Computer 21(3):105–117. https://doi.org/10.1109/2.36
Liu L, Kang Z, Tian L, Xu W, He X (2021) Multilayer graph contrastive clustering network. CoRR. arXiv:2112.14021
Liu Y, Pan S, Jin M, Zhou C, Xia F, Yu PS (2021) Graph self-supervised learning: a survey. CoRR. arXiv:2103.00111
Ma Y, Wang S, Aggarwal CC, Yin D, Tang J (2019) Multi-dimensional graph convolutional networks. In: Proceedings of the 2019 Siam international conference on data mining. SIAM, pp 657–665
Manchanda S, Zheng D, Karypis G (2021) Schema-aware deep graph convolutional networks for heterogeneous graphs. CoRR. arXiv:2105.00644
Mavromatis C, Karypis G (2021) Hemi: multi-view embedding in heterogeneous graphs. CoRR. arXiv:2109.07008
McInnes L, Healy J, Melville J (2018) UMAP: uniform manifold approximation and projection for dimension reduction. https://doi.org/10.48550/ARXIV.1802.03426
Park C, Kim D, Han J, Yu H (2019) Unsupervised attributed multiplex network embedding. CoRR. arXiv:1911.06750
Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Macskassy SA, Perlich C, Leskovec J, Wang W, Ghani R (eds) Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp 701–710
Robinson J, Chuang C, Sra S, Jegelka S (2020) Contrastive learning with hard negative samples. CoRR. arXiv:2010.04592
Shanthamallu US, Thiagarajan JJ, Song H, Spanias A (2020) GrAMME: semisupervised learning using multilayered graph attention models. IEEE Trans Neural Netw Learn Syst 31(10):3977–3988. https://doi.org/10.1109/TNNLS.2019.2948797
Shi S, Xie P, Luo X, Qiao K, Wang L, Chen J, Yan B (2021) Adaptive multi-layer contrastive graph neural networks. CoRR. arXiv:2109.14159
van der Maaten L, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9:2579–2605
Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Lu, Polosukhin I (2017) Attention is all you need. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds) Advances in neural information processing systems, vol 30. Curran Associates Inc, Red Hook
Velickovic P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph attention networks. In: Proceedings of the 6th international conference on learning representations (ICLR)
Wang X, Ji H, Shi C, Wang B, Cui P, Yu PS, Ye Y (2019) Heterogeneous graph attention network. CoRR. arXiv:1903.07293
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 (2020) Deep graph library: a graph-centric, highly-performant package for graph neural networks. CoRR arXiv:1909.01315
Wang X, Liu N, Han H, Shi C (2021) Self-supervised heterogeneous graph neural network with co-contrastive learning. CoRR. arXiv:2105.09111
Xie Y, Zhang Y, Gong M, Tang Z, Han C (2020) MGAT: multi-view graph attention networks. Neural Netw 132:180–189. https://doi.org/10.1016/j.neunet.2020.08.021
Xiong H, Yan J, Pan L (2021) Contrastive multi-view multiplex network embedding with applications to robust network alignment. In: Zhu F, Ooi BC, Miao C (eds) KDD '21: The 27th ACM SIGKDD conference on knowledge discovery and data mining, virtual event, Singapore, August 14–18, 2021. ACM, pp 1913–1923. https://doi.org/10.1145/3447548.3467227
Yang G, Kang Y, Zhu X, Zhu C, Xiao G (2021) Info2vec: an aggregative representation method in multi-layer and heterogeneous networks. Inf Sci 574:444–460. https://doi.org/10.1016/j.ins.2021.06.013
Zangari L, Interdonato R, Calió A, Tagarelli A (2021) Graph convolutional and attention models for entity classification in multilayer networks. Appl Netw Sci 6(1):87. https://doi.org/10.1007/s41109-021-00420-4
Zeng H, Zhou H, Srivastava A, Kannan R, Prasanna V (2019) GraphSAINT: graph sampling based inductive learning method. https://doi.org/10.48550/ARXIV.1907.04931
Zhang C, Song D, Huang C, Swami A, Chawla NV (2019) Heterogeneous graph neural network. In: Teredesai A, Kumar V, Li Y, Rosales R, Terzi E, Karypis G (eds) Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining, KDD 2019, Anchorage, AK, USA, August 4–8, 2019. ACM, pp 793–803. https://doi.org/10.1145/3292500.3330961
Zhao J, Wang X, Shi C, Liu Z, Ye Y (2020) Network schema preserving heterogeneous information network embedding. In: Bessiere C (ed) Proceedings of the twenty-ninth international joint conference on artificial intelligence, IJCAI 2020, pp 1366–1372. ijcai.org. https://doi.org/10.24963/ijcai.2020/190

Publisher's Note