Multi-Type Clustering and Classification from Heterogeneous Networks

Gianvito Pio[a], Francesco Serafino[a], Donato Malerba[a,b], Michelangelo Ceci[a,b]

[a]*Department of Computer Science, University of Bari Aldo Moro, Bari, Italy.*
[b]*Big Data Laboratory, National Interuniversity Consortium for Informatics (CINI), Rome, Italy.*
E-mail: name.surname@uniba.it

Abstract

Heterogeneous information networks consist of different types of objects and links. They can be found in several social, economic and scientific fields, ranging from the Internet to social sciences, including biology, epidemiology, geography, finance and many others. In the literature, several clustering and classification algorithms have been proposed which work on network data, but they are usually tailored for homogeneous networks, they make strong assumptions on the network structure (e.g. bi-typed networks or star-structured networks), or they assume that data are independently and identically distributed (i.i.d.). However, in real-world networks, objects can be of multiple types and several kinds of relationship can be identified among them. Moreover, objects and links in the network can be organized in an arbitrary structure where connected objects share some characteristics. This violates the i.i.d. assumption and possibly introduces autocorrelation. To overcome the limitations of existing works, in this paper we propose the algorithm HENPC, which is able to work on heterogeneous networks with an arbitrary structure. In particular, it extracts possibly overlapping and hierarchically-organized heterogeneous clusters and exploits them for predictive purposes. The different levels of the hierarchy which are discovered in the clustering step give us the opportunity to choose either more globally-based or more locally-based predictions, as well as to take into account autocorrelation phenomena at different levels of granularity. Experiments on real data show that HENPC is able to significantly outperform competitor approaches, both in terms of clustering quality and in terms of classification accuracy.

Keywords: Heterogeneous networks, multi-type clustering, multi-type classification.

1. Introduction

Many objects and data in the real world can be considered interconnected (i.e., through relationships, interactions, etc.), forming complex information networks. Information networks can be found in several social, economic and scientific fields, ranging from the Internet to social sciences, including biology, epidemiology, geography, finance and many others. Current studies about mining networked data mainly focus on homogeneous information networks [31, 28, 41], i.e., networks composed of a single type of object and a single type of link. However, in real life scenarios, there could be multiple types of object, connected to each other through different kinds of link, forming heterogeneous information networks. Examples can be found in biology, where genes, proteins, organisms, etc. are associated to each other through different types of

link. Another example is in the analysis of bibliographic data, where different types of link exist among authors, conferences, journals and papers.

These considerations motivate the recent interest in mining heterogeneous information networks, which in most cases focus on the clustering task. Typically, (*multi-type*) clustering is based on both the attribute values of the objects (possibly of different types) and the links among them (e.g., spectral clustering [26], LinkClus [49] and CrossClus [50]). Clustering has been also exploited as a preliminary phase for other data mining tasks, such as link prediction [3] and semantic tagging [46], as well as for the construction of higher-level features or multiple views of the data [8]. Further interesting examples can be found in studies that proposed ranking-based clustering approaches (e.g., RankClus [44] and NetClus [45]), that generate efficiently results for both ranking and clustering. Some other methods perform classification [18, 19]. They take advantage of links in heterogeneous information networks to propagate knowledge across the nodes, enforcing the similarity between similar objects, if linked. Approaches that work in this direction are based on label propagation [51] and collective classification [39].

One of the main problems that these approaches tackle when learning predictive models from network data (regardless of the type of network, that is, homogeneous or heterogeneous) is that data are affected by some form of autocorrelation [1, 41]. This means that the value of an attribute at a given node depends on the values of the same attribute of the nodes it is connected with. This phenomenon is a direct violation of the assumption that data are independently and identically distributed, which is at the basis of most data mining methods. At the same time, autocorrelation also offers a unique opportunity to improve the performance of predictive models on network data, since inferences about one object can be used to improve inferences about related objects. Autocorrelation can be recognized in several fields, for instance, in spatial data analysis it can be recognized in the (Tobler's) first law of geography: "Everything is related to everything else, but near things are more related than distant things". In social analysis, autocorrelation can be recognized in the homophily principle [29], which shows that people connected through friendship relationships tend to share many sociodemographic, behavioral, and intra-personal characteristics. This is also important in marketing [9].

In this paper, we propose a method which is able to perform both clustering and classification tasks on heterogeneous networks. In particular, it is able to group together heterogeneous objects in a network and to assign labels to unlabeled objects, implicitly taking into account autocorrelation in a collective learning setting. Similarly to multi-type clustering, where cluster labels are associated to objects of multiple types in an unsupervised fashion, in our work we simultaneously cluster and classify objects of different types. Classification can be performed according to different classification schemes i.e. it can be single-label, multi-label, hierarchical, hierarchical multi-label (HMC), etc. In this work we focus attention on the single-label setting and we call the considered learning task *multi-type clustering and classification* from heterogeneous networks. This task is not completely new in the literature and has connections with the task of multiple predicate learning [36] in ILP. The difference is that it is applied to heterogeneous networks and not to logic clauses. Connections can also be found with the task of multi-label collective classification [23, 38], where, however, objects to be classified are of the same type.

In this work, we consider the *within-network* setting [11]: objects for which the class is known are linked to objects for which the class must be estimated [28]. This setting is semi-supervised and differs from the *across-network* setting, where learning is performed from one (labeled) network and prediction is performed on a separate, presumably similar network [27, 41].

In order to simultaneously consider the clustering and the classification tasks, the solution we propose is based on predictive clustering [5], which combines elements from both tasks and

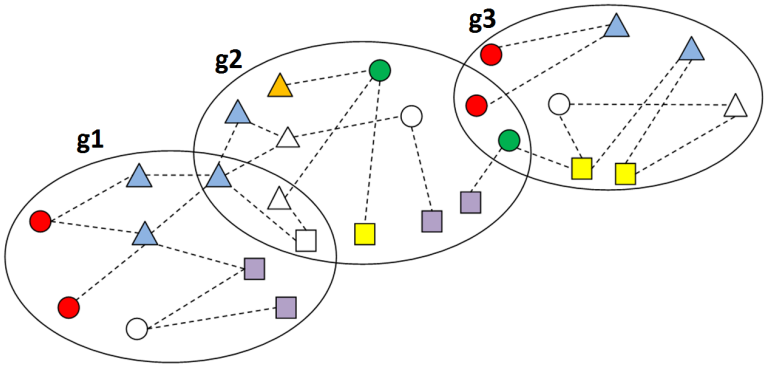

Figure 1: Multi-type classification from heterogeneous networks. Shapes indicate types of object, while colors indicate class labels. White objects are unclassified objects. Ellipses represent possible clusters.

allows us to properly take into account the autocorrelation phenomenon: Clusters of similar objects are identified, and a cluster description and a predictive model are associated to each cluster. Unlabeled objects are assigned to clusters on the basis of the cluster descriptions and the corresponding predictive models are considered to provide predictions for the target property. The basic idea is to build (possibly overlapping) clusters of heterogeneous nodes of the network, such that autocorrelation can be implicitly considered when learning the classifier. This means that the clustering algorithm should preserve the network structure, that is, linked objects should have similar cluster membership [43]. In other words, highly connected objects should fall in the same clusters, contributing to learning the same predictive models. When exploiting multi-type clustering for (multi-type) classification, we use the intuition that class values of objects of type A are in some way related to class values of objects of type B, when all these objects belong to the same heterogeneous cluster (or sub-network). See Figure 1 for an example.

The way the clusters are generated implicitly influences prediction. In fact, a hierarchical organization of clusters, which is learned in this work, facilitates, from the descriptive perspective, the understanding of the results by human experts. From the predictive perspective, this organization results in the possibility of choosing either more globally-based or more locally-based predictions. This is because each cluster can naturally consider different effects of the autocorrelation phenomena on different portions of the network: at higher levels of the hierarchy, clusters will be able to consider autocorrelation phenomena that are spread all over the network, while at lower levels of the hierarchy, clusters will consider the local effects of autocorrelation.

Another characteristic that can influence the prediction is the extraction of overlapping clusters. Indeed, overlapping clusters give the opportunity to base predictions on more than one heterogeneous sub-network. For this reason, we extract hierarchically organized and possibly overlapping heterogeneous clusters.

Since clusters are used for predictive purposes, a clustering algorithm which takes as input the number K of clusters would affect the predictive capabilities of the learned models. In particular, setting K equal to the number of classes would lead to identifying models which are too general and are not able to represent the underlying distribution of the data properly. On the contrary, overestimating the value of K could lead to overfitting problems. Therefore, our clustering algorithm does not require K as input and automatically identifies (at distinct levels of the hierarchy) the optimal number of clusters, on the basis of the data distribution.

The paper, which provides full details about the method and its evaluation, is organized as follows. In the next section, related work is discussed. The method is presented in Section 3, where we formally introduce the problem and describe in detail the proposed predictive clustering-based solution. The experimental results, for several real datasets, are presented in Section 4. Section 5 concludes the paper.

2. Related Work

The motivations for this work come from research reported in the literature in mining heterogeneous network data for both clustering and predictive purposes. In the following subsections, we discuss related work from both research fields, also considering previously proposed predictive clustering methods.

2.1. Clustering heterogeneous network data

The problem of clustering objects in heterogeneous networks has received increasing attention in the last years. One of the first attempts in this direction is presented in [47], where the authors propose improving the cluster quality of interrelated data objects through an iterative reinforcement clustering process. This process aims at iteratively partitioning objects of each type into a given number of clusters. At each iteration, the similarity between two objects a and b of the same type is the linear combination of *i)* the attribute similarity between a and b, *ii)* their intra-type similarity (between the adjacency vectors of a and b, representing links with objects of the same type) and *iii)* their inter-type similarity (between the adjacency vectors of a and b, representing links with objects of different types). Contrary to our task, extracted clusters are not heterogeneous, ignoring the possibility to preserve links among objects of different types.

The authors of [7] propose applying a clustering algorithm in a data warehousing framework. In this case, there are objects of a "central" type (stored in the fact table), which are connected to objects of other types (stored in dimensional tables), forming a star structure. The goal is to build a hierarchy of clusters for each dimensional table, by taking into account both the objects to be clustered and the objects of the central type. These hierarchies are then used for roll-up and drill-down OLAP operations. Since clusters are identified for each dimensional table independently, as in [47], they are not heterogeneous.

The idea of performing clustering of heterogeneous objects organized in a star structure is also considered in [44], where the authors propose RankClus, a probabilistic approach to multi-type clustering. The algorithm assumes that, within each cluster, objects from the central type are generated by a ranking-based probabilistic generative model. The algorithm aims at identifying such models and at exploiting them, in order to compute the posterior probability that each object belongs to each cluster. A limitation of this algorithm is that, although it is possible to extend its application to arbitrary multi-typed star-structured networks, it is tailored for (and experiments are limited to) bi-typed networks. Moreover, the extracted clusters only contain one type of object (i.e., they are homogeneous).

Bi-typed networks are also the main focus of the work presented in [33, 34], where the authors propose the co-clustering method HOCCLUS2 for the analysis of microRNA-gene interaction networks. The main advantages of HOCCLUS2, with respect to the system proposed in [44], are the possibility to group heterogenous objects in the same cluster (thus, preserving links among objects of different types) and the identification of a hierarchy of (possibly overlapping) clusters, which allows an analysis at different levels of granularity. Although similar to the work presented

in this paper, HOCCLUS2 is not able to analyze more than two types of objects; it cannot exploit information conveyed by attributes of objects, and it is not exploited for prediction purposes.

Another work that extracts heterogeneous clusters which preserve links among objects of different types is presented in [13], where the authors propose extending co-clustering to star-structured networks. The method models the problem as the fusion of pair-wise co-clustering sub-problems. However, the star structure limits the types of link that can be considered to those involving one object of the central type. Thus, links among objects not belonging to the central type and links among objects of the same type are not considered and preserved.

In a more recent work [45], the authors present the algorithm NetClus, which takes advantage of the same ranking-based models exploited by RankClus. The main differences are that NetClus can be directly applied to star-structured networks with an arbitrary number of types and that it is able to extract heterogeneous clusters. As in [13], this algorithm suffers from the problem that only links involving one object of the central type can be considered.

The task of *multi-view clustering* on star-structured data [16] also performs clustering from heterogeneous objects. However, this task aims at performing co-clustering on objects of the central type and on features by exploiting multiple (possibly independent) views of the star schema (*á la* co-training).

One of the first approaches where the single central type is not imposed is proposed in [26], although experiments are limited to datasets with such a structure. In this case, the proposed solution is based on spectral clustering and resorts to the relational data mining framework, since it views the network as a relational database. The authors call the task they solve *collective clustering on multi-type relational data*, which has several connections with our clustering task. An important difference is that, as in [47], clusters are not heterogeneous. In general, the task of clustering in relational data mining [25, 30, 20] has some connections with our work because, as stated previously, one can view a relational database as a heterogeneous network. However, these algorithms cluster objects of a single table ("target" table) of the processed database, also on the basis of the objects belonging to the other tables ("non-target" tables). This not only means that extracted clusters are not heterogeneous, but also that only objects of a single type are clustered.

More recently, in [43] the authors propose a method which, similarly to ours, extracts heterogeneous clusters from heterogeneous information networks. The clustering model is probabilistic and is based on the learned strengths of different types of link. Clusters are obtained by an iterative approach, in which the strengths of the links and the quality of the clustering result mutually enhance each other. As in our approach, clusters are overlapped. However, differently from ours, this algorithm takes as input the number of clusters and does not generate hierarchically organized clusters. The implications of these differences have been explained in Section 1.

2.2. Network data classification

In the literature, several approaches have been designed to model a partially labeled network and to provide accurate estimates of labels for unlabeled nodes. These approaches have been studied in the research fields of collective inference [28, 12, 39, 17], active inference [4], semi-supervised and transductive inference [52, 37, 2].

All these approaches, however, are designed to work with homogeneous networks. Only recently, some works started to consider the heterogeneity of nodes and links in the networks. For instance, in [14] the authors work on documents represented as a heterogeneous network consisting of interlinked multimedia objects (containing titles, descriptions and subtitles) and transform such a network into bag-of-words vectors (propositionalization). However, like most

methods based only on propositionalization, it can possibly lose relevant relationships, due to the necessity to apply aggregation functions to attributes associated to sets of linked objects. On the contrary, in [19] the authors work directly on the heterogeneous network and consider a transductive classification task. In particular, they propose a graph-based regularization framework, called GNetMine, which models the link structure in arbitrary information networks. GNetMine considers each relation graph associated to each type of link separately and aims at preserving its consistency. However, GNetMine considers only links among objects (i.e., it is not able to take into account possible attributes associated to nodes), and class labels are associated to heterogeneous sub-networks. This means that the set of the possible class values is common among all the objects, independently of their types. Although such a characteristic may appear reasonable in many domains, it cannot model those (more general) situations in which different classification schemes should be defined for each type of object.

The method proposed in [18] combines ranking and classification tasks on the basis of the intuition that highly-ranked objects within a class should play more important roles in classification or, vice versa, that class membership information is important for determining a good ranking over a dataset. Accordingly, a ranking-based iterative classification framework, called RankClass, is proposed. At each iteration, a graph-based ranking model is built and, on the basis of the current ranking results, the graph structure is adjusted, so that weights of the links in the subnetwork corresponding to each specific class are strengthened, while weights of the links in the rest of the network are weakened. Although experiments show the advantages of combining ranking and classification, as in [19], a single classification scheme can be associated to all the types of object. Moreover, in RankClass, each node is associated to one class, meaning that it is mandatory to associate a classification scheme to each object type.

Recently, in [22] the authors proposed a collective classification approach which aims at classifying objects of the same type in a heterogeneous network, based on the concept of meta-path. A meta-path is a path, between two objects to be classified, consisting of a sequence of link types. This concept is used to effectively assign labels to a group of interconnected instances, by taking into account different meta-path-based dependencies. The classification is probabilistic and is based on the feature values of the object to be classified, on the meta-paths, on the "relational features" associated to the meta-paths, as well as on the labels associated to the objects traversed in the meta-paths. In this way, the proposed model is able to capture the subtlety of different dependencies among instances, with respect to different meta-paths and, at the same time, to implicitly take autocorrelation into account. Although similar to ours, this approach is specifically designed to classify objects of a single type, similarly to the classical classification problem in relational data mining. In the latter context, many approaches have been proposed in the literature, which can be considered relevant baselines for the multi-type classification task. For example, in [6] the authors propose exploiting the naïve Bayes classification method in the multi-relational setting. This system exploits: *i)* first-order classification rules for the computation of the posterior probability for each class; *ii)* both discrete and continuous attributes by applying a supervised discretization method; *iii)* knowledge on the data model (i.e., the database schema) during the generation of classification rules. More recently, the work in [48] introduced the tool RelWEKA, which extends the WEKA toolkit with the multi-relational version of many classical data mining algorithms (e.g., k-NN and SVM).

The adaptation of these approaches to the multi-type classification task consists in the generation of several separate classification models, without exploiting possible dependencies among class values of objects of different types.

2.3. Predictive clustering for network prediction

Some recent works propose the use of predictive clustering in the context of network data. However, none of them consider the heterogeneity of the network.

In [40] the authors follow an approach similar to predictive clustering. In particular, they combine a descriptive task (clustering) with a predictive task (regression) and argue that using networks as a data representation provides a unified framework for identifying and characterizing patterns in climate data. In that case, the network is built a-posteriori on the basis of the values of Pearson's correlation coefficient, computed between pairs of nodes on the time series collected for the same variable. Clustering then boils down to a community detection problem that allows the proposed approach to group interconnected nodes, so that the pairwise walking distance between two nodes in the cluster is minimized. Finally, the prediction of the value of each node is obtained by means of a linear regression model built on the (spatial) cluster it belongs to. The innovative aspects of our proposal with respect to this work are twofold. First, extracted clusters are overlapping and hierarchically organized. Second, the correlation is measured on nodes which are interconnected in an existing network, while the method in [40] measures the correlation to define virtual edges between examples.

In [41] the authors propose learning network predictive clustering trees, in order to classify nodes in a network. Similarly to classical decision trees, network predictive clustering trees are built top-down. The difference is that the authors propose using a network autocorrelation-based measure as a heuristic, thus explicitly taking autocorrelation into account. The same principle is adopted in [42], where network predictive clustering trees are used to predict gene functions. The difference is that, in this last work, a hierarchical multi-label classification scheme is considered.

3. Multi-type Clustering and Classification from Heterogeneous Networks

In this section, we first introduce notations, definitions and concepts for both clustering and classification tasks in heterogeneous information networks. Subsequently, we present our solution for the identification of the strength of the relationships among nodes of target types, and our method to identify clusters and organize them hierarchically. Finally, we present the classification step. The first definition we introduce is of the heterogeneous information network:

Definition 1 (Heterogeneous information network). *A heterogeneous information network is an information network where objects and links can be of different types. It is represented as an undirected graph $G = (V, E)$, where V is the set of nodes (i.e., objects) and E is the set of edges (i.e., links) between objects. Moreover:*

- *each node $v' \in V$ is associated to a single node type $t_v(v') \in \mathcal{T}$, where $\mathcal{T}$ is the finite set $\{T_p\}_p$ of all the possible types of node in the network;*
- *each node type T_p defines a subset of nodes $V_p \subseteq V$;*
- *each node type T_p defines a set of attributes $\mathcal{X}_p = \{X_{p,1}, \ldots, X_{p,m_p}\}$ that are associated with the nodes of that type;*
- *each edge between two nodes v' and v'' is associated with an edge type $R_j \in \mathcal{R}$, where $\mathcal{R}$ is the finite set $\{R_j\}_j$ of all the edge types in the network. Formally, we define an edge as $e = \langle R_j, \langle v', v'' \rangle \rangle \in E$, where $R_j = t_e(e) \in \mathcal{R}$ is its type;*

- *each edge type R_j defines a subset of edges $E_j \subseteq (V_p \times V_q)$, where $T_p \leq T_q$*[1]*;*
- *the set of node types $\mathcal{T}$ is partitioned into $\mathcal{T}_t$ (target types), which are considered as the target of the clustering/classification task, and $\mathcal{T}_{tr}$ (task-relevant types), consisting of the types that are not the main subject of the analysis. Only nodes belonging to a type of $\mathcal{T}_t$ are actually clustered and possibly classified, on the basis of the edges involving nodes of any type in $\mathcal{T}$.*

Definition 2 (Heterogeneous cluster). *Given a heterogeneous network $G = (V, E)$, we define a heterogeneous (or multi-type) cluster as $G' = (V', E')$, where:*

- *$V' \subseteq V$;*
- *$\forall v' \in V', t_v(v') \in \mathcal{T}_t$, that is, nodes in the clusters only belong to target types;*
- *$E' \subseteq (E \cup \hat{E})$ is a set of edges among the nodes in V', belonging either to the set of edges E or to a set of extracted edges $\hat{E}$, that are edges identified by the clustering method (details about these edges will be discussed in Section 3.1).*

Definition 3 (Hierarchical organization of clusters). *A hierarchy of heterogeneous clusters is defined as a list of hierarchy levels $[L_1, L_2, \ldots, L_k]$, each consisting of a set of heterogeneous and overlapping clusters. For each level $L_i, i = 2, 3, \ldots . k$, we have that $\forall\ G' \in L_i\ \exists\ G'' \in L_{i-1}$, such that G'' is a subgraph of G'.*

Definition 4 (Classification schemes in heterogeneous networks). *Given a heterogeneous network $G = (V, E)$, each target type $T_p \in \mathcal{T}_t$ can be associated to a different classification scheme $S_p \in \mathcal{S}$, where $\mathcal{S}$ is the set of classification schemes. More formally, we define a partial mapping function $\phi : \mathcal{T}_t \rightarrow \mathcal{S}$ which maps each node type T_p to the classification scheme S_p. As stated before, in this work we focus on single-label schemes. The mapping function ϕ is partial in the sense that some target types may have no classification scheme. In other words, nodes belonging to some types could be considered as the target only for the clustering task, but not for classification purposes. In this respect, we define the set of types $\mathcal{T}_t^* \subseteq \mathcal{T}_t$ as the set of target types which also have an associated classification scheme. Therefore, each target type in $\mathcal{T}_t^*$ will have a single target attribute, whose value will be predicted for unlabeled nodes.*

On the basis of these definitions (see Appendix for a compact view of the symbols), we define the learning task we consider in this work.

Definition 5 (Predictive hierarchical clustering for multi-type classification). *Given a heterogeneous network $G = (V, E)$, the set of target types T_t and the partial mapping function $\phi : \mathcal{T}_t \rightarrow \mathcal{S}$, the goal is to find:*

- *A hierarchy of heterogeneous clusters $[L_1, L_2, \ldots, L_k]$.*
- *A classification function $\psi_p^{(i)} : V_p \rightarrow \mathcal{Y}_p$ for each level L_i and for each target type $T_p \in \mathcal{T}_t^*$. $\mathcal{Y}_p$ is defined according to the classification scheme $S_p = \phi(T_p)$.*

[1] Since we consider an *undirected* graph representation, we define an ordering among the node types. In this way, each edge type R_j defines a subset of links E_j which associate pairs of nodes belonging to node types T_p and T_q, where T_q follows T_p according to a total ordering. This guarantees that the edge $\langle v', v'' \rangle$ does not appear also in the form $\langle v'', v' \rangle$.

The different classification functions identified for different levels of the hierarchy are exploited to perform either globally-based or locally-based predictions, taking into account autocorrelation phenomena on different portions of the network.

The method we propose for the task in Definition 5 is based on four steps:

1. Identification of the strength of the relationship among nodes;
2. Extraction of a set of (possibly overlapping) multi-type clusters in the form of multi-type cliques (see Section 3.2), which are not hierarchically organized;
3. An iterative process in which pairs of multi-type clusters are merged when some heuristic criteria are satisfied. At each iteration, several pairs of clusters can be merged. The process stops when no changes are produced in the last iteration, otherwise an additional level of the hierarchy is added.
4. Identification of a classification function for each hierarchical level and for each target type with an associated classification scheme. Each function associates each node of a given target type to a value in the range defined by the classification scheme associated to such a target type.

For the sake of simplicity, hereafter we will use the terms *clique* and *cluster* to refer to multi-type clique and multi-type cluster, respectively.

3.1. Computing the strength of the relationship among nodes of different types

In this subsection we clarify how we compute the strength of the relationship among the nodes. In order to explain this concept, we first provide the definition of tuples of nodes, which is used to represent the output of this phase.

Definition 6 (Tuple of nodes). *A tuple is defined as* $\vec{v_i} \in T_{p_1} \times T_{p_2} \times \ldots \times T_{p_r}$*, where* $\forall j = 1, \ldots, r$*:* $T_{p_j} \in \mathcal{T}_t$ *and* $\mathcal{T}_t - \{T_{p_1}, T_{p_2}, \ldots T_{p_r}\} = \emptyset$*. In other words, the structure of a tuple is represented by all the target types, taken at least once (i.e.,* $r \geq |\mathcal{T}_t|$*)*[2].

Given the structure of the tuple, for each possible tuple $\vec{v_i}$, it is possible to compute the score s_i which represents the strength of the relationship among all the nodes in the tuple. The way we identify this value takes into account the possible meta-paths[3] in the network that pairwise connect all the nodes in the tuple, i.e.: $s_i = \min_{j' \neq j''} score(v_{i,j'}, v_{i,j''})$, where $v_{i,j}$ is the j-th node of the tuple $\vec{v_i}$. If there is no meta-path connecting two nodes in the tuple, $s_i = 0$. The rationale behind the use of the minimum is that the connection among the nodes in the tuple is strong if all the pairwise connections are strong (see Figure 2).

Accordingly, we define the set of *extracted* edges $\hat{E}$, which consists of the edges for each pair of nodes $v_{i,j'}$,$v_{i,j''}$ not belonging to E for which $score(v_{i,j'}, v_{i,j''}) > 0$. Each edge between two nodes $v_{i,j'}$ and $v_{i,j''}$ is associated to the maximum score computed according to the meta-paths connecting them in the network:

$$score(v_{i,j'}, v_{i,j''}) = \max_{P \in metapaths(T_{j'}, T_{j''})} pathscore(P, v_{i,j'}, v_{i,j''}), \quad (1)$$

[2]It is noteworthy that, although we do not impose any limit on the value of r, it will be a predefined (very low) value, defined according to the set of target types considered for the clustering and classification tasks. Therefore, the number of possible tuples is always finite.

[3]In this paper, the term *meta-path* represents a concept similar to the *meta-path* in [22], where it represents a sequence of link types. The difference is that, in our paper, a *meta-path* represents the set of sequences of nodes which follow the same sequence of link types. Accordingly, the length of a *meta-path* is the number of link types in the sequence.

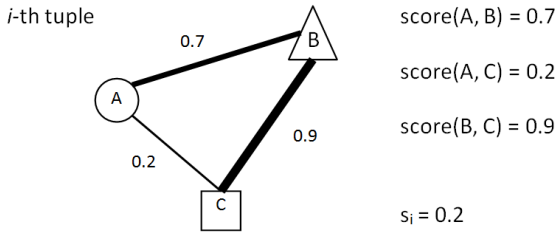

Figure 2: Strength of the relationships for the tuple $\vec{v}_i = \langle A, B, C \rangle$.

#Seq	A_id	A_att1	A_att2	B_id	B_att1	B_att2
1	t101	Tom Ford	UK	t201	0,6	0,15
2	t101	Tom Ford	UK	t204	1,0	0,21
3	t102	Mike Walls	New York	t209	0,2	0,18
4	t103	Paul Morgan	Germany	t203	0,8	0,27
5	t103	Paul Morgan	Germany	t210	0,4	0,27
6	t103	Paul Morgan	Germany	t204	0,8	0,20
7	t104	Paolo Belli	Italy	t209	0,6	0,40

Figure 3: An example of analysis of the sequences associated to the nodes $v_{i,j'}$ (with id = *t103*) and $v_{i,j''}$ (with id = *t209*). In the example, sequences 4, 5 and 6 (in yellow) are associated to $v_{i,j'}$, and sequences 3 and 7 (in green) are associated to the node $v_{i,j''}$. The algorithm pair-wisely compares the two sets of sequences (sequences in yellow and sequences in green) and computes the similarity between $v_{i,j'}$ and $v_{i,j''}$ as the maximum similarity between two sequences.

where:

- $metapaths(T_{j'}, T_{j''})$ is the finite set $\{P_1, P_2, \ldots, P_c\}$ of all the possible meta-paths between the nodes of type $t_v(v_{i,j'}) = T_{j'}$ and the nodes of type $t_v(v_{i,j''}) = T_{j''}$. We consider the c shortest meta-paths;
- each meta-path P is represented as a finite set of sequences of nodes $\{seq_d = \langle v_1^{(d)}, v_2^{(d)}, \ldots, v_z^{(d)} \rangle\}$, such that: *i)* the i-th node of each sequence in the meta-path P is of the same type; *ii)* the first and the last nodes are of type $T_{j'}$ and $T_{j''}$, respectively and *iii)* two consecutive nodes in a sequence are connected in E;
- $pathscore(P, v_{i,j'}, v_{i,j''})$ represents the strength of the connection between $v_{i,j'}$ and $v_{i,j''}$ in P.

If there is a sequence in P that connects $v_{i,j'}$ and $v_{i,j''}$, $pathscore(P, v_{i,j'}, v_{i,j''}) = 1$. Otherwise, it is computed as the maximum similarity between the sequences which start with $v_{i,j'}$ and the sequences which end with $v_{i,j''}$ (see Figure 3). Formally:

$$pathscore(P, v_{i,j'}, v_{i,j''}) = \max_{\substack{seq_{d'}, seq_{d''} \in P; \\ v_1^{(d')} = v_{i,j'} \; ; \; v_z^{(d'')} = v_{i,j''}}} SS(seq_{d'}, seq_{d''}) \tag{2}$$

In this formula, the similarity between two sequences ($SS(\cdot,\cdot)$) is computed as the average between the similarity of target attribute values and the similarity of non-target attribute values of the nodes in the two sequences, that is:

$$SS(seq_{d'}, seq_{d''}) = \frac{1}{2} \cdot \left[\frac{\sum_{x \in X_C(seq_{d'}, seq_{d''})} s_x(seq_{d'}, seq_{d''})}{|X_C(seq_{d'}, seq_{d''})|} + \frac{\sum_{x \in X_{NC}(seq_{d'}, seq_{d''})} s_x(seq_{d'}, seq_{d''})}{|X_{NC}(seq_{d'}, seq_{d''})|} \right], \quad (3)$$

where:

- $X_C(seq_{d'}, seq_{d''})$ is the set of target attributes;
- $X_{NC}(seq_{d'}, seq_{d''})$ is the set of non-target attributes;
- $s_x(seq_{d'}, seq_{d''})$ is the similarity between $val_x(seq_{d'})$, that is the value of the attribute x in the sequence $seq_{d'}$, and $val_x(seq_{d'})$, that is the value of the attribute x in the sequence $seq_{d''}$.

Following [15], $s_x(seq_{d'}, seq_{d''})$ is computed as follows:

- if x is numeric, then $s_x(seq_{d'}, seq_{d''}) = 1 - \frac{|val_x(seq_{d'}) - val_x(seq_{d''})|}{max_x - min_x}$, where min_x (resp. max_x) is the minimum (resp. maximum) value, for the attribute x;
- if x is not a numeric attribute, then $s_x(seq_{d'}, seq_{d''}) = 1$ if $val_x(seq_{d'}) = val_x(seq_{d''})$, 0 otherwise.

If at least one value is not available (i.e., is *null*), the attribute is ignored. Moreover, the same attribute can be considered more than once in the sequences. This can happen because nodes of the same type can be considered in the same sequence, taking into account the possible presence of network autocorrelation.

Finally, some node types may not be involved in any meta-path. In order to exploit the information conveyed by these nodes, we add an aggregation of their attribute values to the nodes that are connected to them and that appear in at least one meta-path (see Figure 4).

Such an aggregation considers values coming from directly or indirectly (up to a predefined *depth* of analysis) connected nodes. For this purpose different aggregation functions could be used. In our case, we use the *arithmetic mean* for numeric attributes and the *mode* for attributes of any other type.

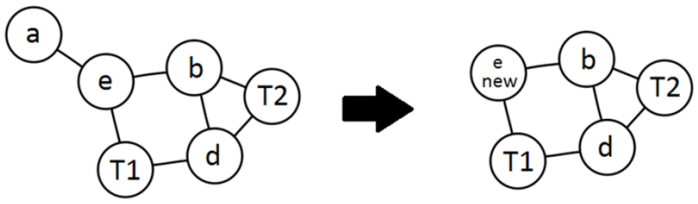

Figure 4: An example of the aggregation of nodes that are not involved in any meta-path. In the example T1 and T2 are target nodes. Since node *a* cannot be involved in any sequence of a meta-path, the aggregation phase builds a new node *e_new*, which aggregates in *e_new* information coming from *a*.

Algorithm 1 Identification of the first hierarchical level L_1.

Require: List of initial clusters L_1 built from $\{\vec{v_i} \,|s_i \geq \beta\}$; the ordering relation $<_c$.
Ensure: The updated list of clusters L_1 (first level) in the form of cliques.

```
cliques ← [ ];
repeat
    sort L1 according to <c;
    mergedClusters ← [ ];
    for i = 1 to |L1| − 1 do
        merged ← false;
        {Search for a merging leading to a clique}
        for j = i + 1 to |L1| do
            if merge(L1[i], L1[j]) is a clique then
                add merge(L1[i], L1[j]) to mergedClusters;
                remove L1[j] from L1; merged ← true;
            end if
        end for
        {If L1[i] cannot be merged, add it to the result}
        if merged = false then
            add L1[i] to cliques;
        end if
    end for
    L1 ← mergedClusters;
until mergedClusters = [ ]
return cliques
```

3.2. Building multi-type cliques

Once all the possible tuples are identified, each associated with a strength score, we build a set of (possibly overlapping) clusters in the form of cliques. A cluster is in the form of a clique if all the tuples involving only nodes in the cluster have a score (strength of the relationships) above a given threshold β. Consequently, the proposed algorithm requires the parameter $\beta \in [0, 1]$, used to consider a relationship relevant or not. The task to be solved is formalized in the following.

Given:

- a finite set of tuples $\{\vec{v_i}\}_i$, each associated with its own score s_i, computed according to the algorithm described in Section 3.1;
- a threshold β;
- a (total) ordering relation $<_c$ between clusters, which should reflect the quality of the clusters (the criteria considered in this paper are described in Section 3.5).

Find: a set of clusters in the form of cliques, according to β and the relation $<_c$.

The algorithm consists of the following steps:

i) A filtering phase on the set of tuples, which keeps only the tuples with a score greater than (or equal to) β. The result is the subset of tuples $\{\vec{v_i} \,|s_i \geq \beta\}$.
ii) The generation of an initial set of cliques, each consisting of a tuple in $\{\vec{v_i} \,|s_i \geq \beta\}$. This is the input for Algorithm 1, which identifies the first hierarchy level L_1.

iii) A process that iteratively merges two clusters G' and G'' into a new cluster G''' (see Algorithm 1). The initial set of clusters is regarded as a list and is sorted according to the ordering relation $<_c$. Then, each cluster G' is merged with the first cluster G'' in the list which leads to a merged cluster G''', which is still a clique. This process is repeated until no more merging takes place.

The result of this phase is a set of clusters which defines the first level of the hierarchy L_1. In the next subsection we explain how we build the other levels.

3.3. Building the hierarchy of clusters

Once the first level L_1 of the hierarchy has been identified, we evaluate whether some pairs of clusters (cliques, in L_1) can be reasonably merged to build the other levels.

The approach is similar to that adopted in Algorithm 1. The main difference is that, instead of working on cliques, it works on generic clusters, where the strength score associated to each tuple is not necessarily greater than β. This means that a different criterion for the identification of candidates for merging has to be considered. In particular, we use a criterion which is inspired by the research in hierarchical co-clustering, where a common stopping criterion is based on a threshold applied to the quality of clusters obtained by merging two clusters [33]. Analogously, in our approach, two clusters G' and G" are merged into a cluster G''' if $h(G''') > \alpha$, where α is a user-defined threshold. Low values of α lead to a higher number of mergings at the price of lower quality of clusters. The task is formalized in the following.

Algorithm 2 Identification of the hierarchy of clusters $L_1, L_2, \ldots L_k$.

Require: The first level of the hierarchy L_1; the ordering relation $<_c$; the heuristic function $h(\cdot)$; the threshold α.
Ensure: The hierarchy of clusters $L_1, L_2, \ldots L_k$.

```
k = 1;
repeat
    L ← L_k; k ← k + 1; clusters ← [ ]; mergedClusters ← [ ];
    sort L according to <_c;
    for i = 1 to |L| − 1 do
        merged ← false;
        {Search for a merging which satisfies the threshold}
        for j = i + 1 to |L| do
            temp ← merge(L[i], L[j]);
            if h(temp) >= α) then
                add temp to mergedClusters;
                remove L[j] from L; merged ← true;
            end if
        end for
        {If L[i] cannot be merged, add it to the result}
        if merged = false then
            add L[i] to clusters;
        end if
    end for
    {Build a new level if there was at least one merging}
    if mergedClusters ≠ [ ] then
        L_k ← clusters;
    end if
until mergedClusters = [ ]
return L_1, L_2, ... L_k
```

Given:

- the first hierarchy level L_1, identified as described in Section 3.2;
- the ordering relation $<_c$;
- a threshold α;
- a heuristic function $h(G)$ which evaluates a cluster G according to a quality criterion (the criterion considered in this paper is described in Section 3.5).

Find: a list of hierarchical level $[L_1, L_2, \ldots, L_k]$, according to Definition 3.

The method, which is formally described in Algorithm 2, builds a new hierarchy level when a further iteration of the *repeat..until* loop is possible. It is noteworthy that L_k (i.e., the top level of the hierarchy) does not necessarily contain a single cluster, meaning that a forest of hierarchies of clusters is actually returned.

3.4. Classification

Once the hierarchy has been identified, we define a classification function $\psi_p^{(i)}$ for each level i and each target type $T_p \in T_t^*$. Unlabeled nodes are classified according to a variant of the majority voting algorithm which considers the overlapping among clusters. In particular, predictive models are built by considering the labels of the nodes of the same type belonging to the clusters in which the node falls.

Keeping in mind that we work in a semi-supervised learning setting, we have to deal with the fact that both labeled and unlabeled nodes, possibly of different types, belong to the same cluster. Obviously, clusters with a high percentage of labeled nodes lead to more reliable predictions. In our approach we consider this aspect and we use the percentage of labeled nodes belonging to each cluster, in order to compute a weight representing the reliability of the prediction model.

Formally, for each hierarchy level L_i, the classification function $\psi_p^{(i)}(v)$ for an unlabeled node v of the target type T_p exploits a weighted average of the probabilities that v belongs to the class Y in a cluster G':

$$P(cl(v) = Y | v \in un(G'_p)) = \frac{freq(Y, lab(G'_p))}{|lab(G'_p)|}, \tag{4}$$

where:

- $cl(v)$ is a possible class (label) to associate with v;
- G'_p is the set of nodes of type T_p in G';
- $lab(G'_p)$ is the set of labeled nodes of type T_p in G';
- $un(G'_p)$ is the set of unlabeled nodes of type T_p in G';
- $freq(Y, lab(G'_p))$ is the number of nodes of type T_p labeled as Y in G'_p.

The classification function $\psi_p^{(i)}(v)$ can now be defined as:

$$\psi_p^{(i)}(v) = \arg\max_{Y \in \mathcal{Y}_p} \frac{\sum_{\{G' | v \in G' \wedge G' \in L_i\}} \left[w(G'_p) \cdot P(cl(v) = Y | v \in un(G'_p)) \right]}{\sum_{\{G' | v \in G' \wedge G' \in L_i\}} w(G'_p)}, \tag{5}$$

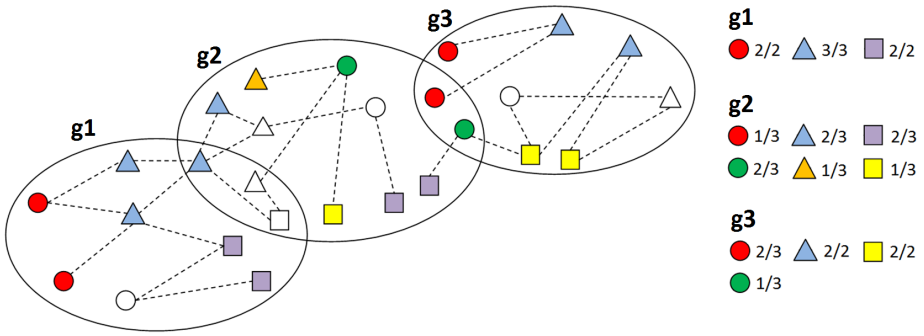

Figure 5: Predictive models associated to each type of node in the heterogeneous network. Shapes indicate types of object, while colors indicate class labels. White objects are unclassified objects. Ellipses represent possible clusters.

where $\{G'|v \in G' \wedge G' \in L_i\}$ is the set of clusters of level L_i which v belongs to, and $w(G'_p) = |lab(G'_p)|/|G'_p|$ is the percentage of labeled nodes of type T_p in G'.

To better explain how classification is performed, we report an example of classification models built for unlabeled nodes of clusters in Figure 5. In this example, an unlabeled circle node belonging only to cluster $g3$ is classified according to the classification model of circle nodes built from $g3$, that is, $[red = 2/3, green = 1/3]$. Therefore, it is classified as *red*. When a node falls in more than one cluster, the classification model is built according to all the clusters involved, according to Equation (5). For example, the white square node v' belonging to both $g1$ and $g2$ is classified according to the weighted average of the classification models $[purple = 1, yellow = 0]$ (cluster $g1$) and $[purple = 2/3, yellow = 1/3]$ (cluster $g2$). Since $g1$ has 2 out of 3 labeled square nodes, whereas $g2$ has 3 out of 4 labeled square nodes, the classification model of $g1$ will be associated to a weight of $2/3$, while that of cluster $g2$ will be associated to the weight $3/4$. According to Equation (5), v' will be classified on the basis of the following model:

$$\psi_p^{(i)}(v') = \underset{purple,yellow}{\arg\max} \left[purple = \frac{2/3 \cdot 1 + 3/4 \cdot 2/3}{2/3 + 3/4}, yellow = \frac{2/3 \cdot 0 + 3/4 \cdot 1/3}{2/3 + 3/4} \right] = purple$$

Here, we assume that unlabeled nodes are known and used during the clustering phase, which is in line with the transductive variant of semi-supervised learning.

Formula (5) allows our method to capture the autocorrelation that relates a target attribute with an independent attribute (relationally-lagged explanatory attributes) for two connected objects, possibly of different types. This form of autocorrelation is more general than that which relates a target attribute with itself (relationally-lagged response attribute). However, since we can consider the same type more than once as the target type in clustering, we can force the method to catch the form of autocorrelation which relates a target attribute with itself. This turns out to be useful also when we have only one target type. To balance the contribution of the two forms of autocorrelation, we introduce the parameter $\gamma \in [0, 1]$. With $\gamma = 0$ we only catch the autocorrelation that relates a target attribute with an independent attribute, whereas with $\gamma = 1$ we only catch the autocorrelation that relates a target attribute with itself.

This means that $\psi_p^{(i)}(v)$ can be rewritten as:

$$\begin{aligned}\psi_p^{(i)}(v) &= \arg\max_{Y \in \mathcal{Y}_p} \Bigg(\gamma \cdot \frac{\sum_{\{G' | v \in G' \wedge G' \in L_i^*\}} \left[w(G'_p) \cdot P(cl(v) = Y | v \in un(G'_p)) \right]}{\sum_{\{G' | v \in G' \wedge G' \in L_i^*\}} w(G'_p)} + \\ &+ (1-\gamma) \cdot \frac{\sum_{\{G' | v \in G' \wedge G' \in L_i\}} \left[w(G'_p) \cdot P(cl(v) = Y | v \in un(G'_p)) \right]}{\sum_{\{G' | v \in G' \wedge G' \in L_i\}} w(G'_p)} \Bigg) \end{aligned} \tag{6}$$

where L_i^* is the set of clusters belonging to the i-th hierarchical level, obtained by running the algorithm considering one single target type twice (i.e., for type T_p, identified tuples consist of two nodes of type T_p). In this way, we catch the autocorrelation between the target attribute (of the nodes of type T_p) and itself.

3.5. The ordering relation $<_c$ and the heuristics $h(\cdot)$

In this subsection, we describe the approach we adopt to sort clusters (i.e., the ordering relation $<_c$) and to select an appropriate candidate for merging (i.e., the heuristic function $h(\cdot)$). Both the ordering relation and the heuristic function aim at the maximization of the strength of the relationships among nodes of different types in the clusters. In this way, we take into account the (possible) dependencies among target attributes of different target types within the same cluster.

The considered measure is inspired by the concept of *cohesiveness*, which is typically used in coclustering algorithms [33] to represent the average strength of the connection among objects of two different types in the same cocluster. In this work, we extend this concept to multiple types and, in particular, we consider the average score of all the possible tuples that can be built from the nodes belonging to a cluster. Formally, we compute $h(G')$ as the *multi-type cohesiveness*:

$$h(G') = \frac{1}{|tuples(G')|} \cdot \sum_{\vec{v_i} \in tuples(G')} s_i \tag{7}$$

where $tuples(G')$ is the set of tuples that can be built from the nodes belonging to G', and s_i is the score associated with each tuple. The ordering relation $<_c$ is then defined on the basis of the heuristic function $h(\cdot)$: $G' <_c G'' \iff h(G') > h(G'')$ that is, one cluster is processed before another if it has a higher cohesiveness.

3.6. Time complexity

The analysis of the time complexity has to take into account the complexity of each single phase, that is, the complexity of the identification of the possible tuples, the complexity of the identification of the first and subsequent levels of the hierarchy of clusters and, finally, the complexity of the classification of unlabeled nodes.
For the identification of possible tuples, we assume that:

- nodes are equally distributed among types, i.e., $|V_1| = |V_2| = \ldots = |V_{|\mathcal{T}|}| = n$;
- we consider at most c meta-paths connecting nodes of two different types;
- each node (of every type) is represented according to m attributes.

In the worst case, a meta-path involves all the types of edge $|\mathcal{R}|$. Therefore, for each pair of target types, the time complexity for the identification of the tuples requires $O(c \cdot 2 \cdot |\mathcal{R}| \cdot n)$ (for the identification of the sequences, when nodes are indexed according to an order-preserving data structure) plus $O(c \cdot n^2 \cdot m \cdot |\mathcal{R}|)$ (for the computation of the similarities among sequences that start with a node of a target type and end with a node of another target type, when there is no sequence that connects the two nodes, see Equation (2)). In these equations, $m \cdot |\mathcal{R}|$ is due to the computation of the similarity. This means that the cost of extracting tuples is: $O(|\mathcal{T}_t| \cdot (|\mathcal{T}_t| - 1)/2 \cdot c \cdot (2 \cdot n \cdot |\mathcal{R}| + n^2 \cdot m \cdot |\mathcal{R}|))$, where $|\mathcal{T}_t| \cdot (|\mathcal{T}_t| - 1)/2$ represents the number of possible pairs of target types. Since $n^2 \cdot m \cdot |\mathcal{R}|$ dominates over $2 \cdot n \cdot |\mathcal{R}|$ and since $|\mathcal{T}_t|$ and c are small constants, the complexity for identifying tuples is:

$$O(n^2 \cdot m \cdot |\mathcal{R}|) \tag{8}$$

As for the identification of the first and subsequent levels of the hierarchy, we have to take into account the number of tuples which, in the worst case, is $O(|\mathcal{T}_t| \cdot (|\mathcal{T}_t| - 1)/2 \cdot \gamma^2) = O(\gamma^2)$, where $\gamma \ll n$ is the average number of objects of a given target type which are connected, according to a meta-path, to a node of another target type.

Therefore, if a cluster contains $O(\gamma^2)$ tuples, each iteration of the *repeat...until* loop of both Algorithms 1 and 2 for the evaluation of $h(\cdot)$ costs $O(\gamma^4)$ (due to the pairwise evaluation of the clusters). By assuming that the number of clusters halves at each iteration (which happens if each cluster can be merged with another cluster), the number of possible iterations is $O(log_2(\gamma^2))$. Therefore, the complexity of Algorithms 1 and 2 can be approximated to:

$$O(\gamma^4 \cdot log_2(\gamma^2)) \tag{9}$$

The classification of unlabeled nodes only requires a scan of the clusters (at all the levels) and the application of the classification algorithm, whose time complexity is linear in the number of nodes that fall in the containing clusters. This means that the complexity of this phase is linear with respect to: the number of hierarchical levels (k), the number of target types with an associated scheme ($|\mathcal{T}_t^*|$), n, the number of clusters and the average number of labeled examples per cluster.

In short, by combining Equations (8) and (9) with the time complexity of the classification phase, we have that the time complexity of the whole algorithm is:

$$O(n^2 \cdot m \cdot |\mathcal{R}| + \gamma^4 \cdot log_2(\gamma^2)) \tag{10}$$

which strongly depends on the value of γ. If the dataset, according to the meta-paths, has a relatively small number of connections (i.e., $\gamma^4 \cdot log_2(\gamma^2) \leq n^2 \cdot m \cdot |\mathcal{R}|$), the time complexity is $O(n^2 \cdot m \cdot |\mathcal{R}|)$. Otherwise, the time complexity is $O(\gamma^4 \cdot log_2(\gamma^2))$.

4. Experiments

The proposed method has been implemented in the system HENPC (HEterogeneous Network Predictive Clustering), which is publicly available at `www.di.uniba.it/~gianvitopio/systems/henpc/`. In this section, we describe the experiments aiming at the evaluation of HENPC for both clustering and classification tasks.

Before presenting the results, in the following subsections, we briefly describe the considered datasets, the competitor systems and the experimental setting.

4.1. Datasets

As for the datasets, we give some details about the number of nodes, the number of edges and the considered target types.

WEBKB. This dataset contains web-pages collected by the World Wide Knowledge Base project of the CMU group[4]. We processed it in order to create a network consisting of 3 node types and 3 edge types. The clustering task focuses on *pages* and *terms*, while the classification task focuses on *pages*, which are classified into: *student*, *faculty* and *course*. The network contains 1,586 nodes and 34,359 edges.

UWCSE. This dataset contains data about the Department of Computer Science and Engineering of the University of Washington[5]. The data include faculty members, their publications, their projects and the courses they teach. Moreover, there is all the required information to describe the relationships among faculty members. We processed the dataset in order to create a network consisting of 7 node types and 7 edge types. The clustering task focuses on *persons* and *courses*, while the classification task focuses on *courses*, which are classified into: *graphics*, *theory*, *ai*, *language* and *systems*. Overall, the network contains 785 nodes and 1,104 edges.

DBLP. This dataset contains data about the co-authorship of scientific papers in computer science stored in the DBLP database. We discarded papers written by a single author. Moreover, we performed an equal-frequency discretization on the number of citations of each paper, obtaining three discrete values: *low*, *medium* and *high*. Finally, we processed the dataset in order to create a network consisting of 3 node types and 2 edge types. Clustering focuses on *papers* and *authors*, while classification focuses on *papers*, which are classified into: *low*, *mid* and *high* (number of citations). The network contains 1,867 nodes and 2,494 edges.

MOVIES. This dataset is built from the MovieLens100k dataset[6] and contains ratings from 1,000 users on 1,700 movies, which are collected by the *movielens* recommender system. We processed the dataset in order to create a network consisting of 4 node types and 3 edge types. The clustering task focuses on *users* and *movies*, while the classification task focuses on *movies*, which are classified into: *comedy*, *thriller*, *drama* and *action*. The network contains 2,051 nodes and 59,532 edges.

YELP. Yelp is a website where it is possible to find, review and talk about business activities. The dataset has been released for academic purposes[7], to supply real-world data for the experimental evaluation of machine learning methods. Data include: businesses, business characteristics, users, check-in information, friendship relationships, tips and reviews. We processed the dataset in order to create a network consisting of 7 node types and 8 edge types. Both clustering and classification focus on *business* and *users*. In particular, we select the top-four categories of businesses, that are: *Restaurants*, *Beauty & Spas*, *Health & Medical* and *Shopping*, whereas users are classified into: *low* and *high*, representing the average rating they gave to businesses. The processed network contains 41,006 nodes and 41,906 edges.

IMDB. This dataset[8] is an extension of the MovieLens10M dataset, published by the GroupLens research group. It links the movies of the MovieLens dataset with their corresponding web pages in Internet Movie Database (IMDb) and Rotten Tomatoes movie review systems. From the dataset, we kept users with both rating and tagging information. Moreover, we processed the

[4] `www.cs.cmu.edu/afs/cs/project/theo-20/www/data/`
[5] `alchemy.cs.washington.edu/data/uw-cse/`
[6] `grouplens.org/datasets/movielens/`
[7] `www.yelp.com/dataset_challenge`
[8] `grouplens.org/datasets/hetrec-2011/`

dataset in order to create a network consisting of 13 node types and 16 edge types. Clustering focuses on *users* and *movies*, while classification focuses on *movies*, which are classified into: *comedy*, *thriller*, *drama* and *action*. The network contains 97,030 nodes and 56,153 edges.

4.2. Experimental setting

In order to evaluate the clusters identified by HENPC, we performed a comparison with the system HOCCLUS2 [33], which also discovers possibly overlapping and hierarchically organized clusters. It is tailored to work with two object types (i.e., it is a co-clustering algorithm) and was originally used in bioinformatics to discover biological correlations between microRNAs and target genes.

In order to perform a fair comparison of the clustering results, we consider an evaluation measure which is different from the optimization criterion adopted by both HENPC and HOCCLUS2. In particular, the identified clusters are evaluated in terms of a variant of the Q-Modularity [32], which measures the quality of the clustering with respect to a random clustering. This variant is described in the following. Let:

$$e(G', G'') = \frac{1}{2|E \cup \hat{E}|} \sum_{\vec{v_i} \in tuples(G', G'')} s_i$$

be a measure of the strength of the edges considering objects in cluster G' and objects in cluster G'', where $tuples(G', G'')$ is the set of tuples that can be built between objects of different types belonging to clusters G' and G''. Intuitively, we aim at clusters for which $e(G', G')$ values are generally large and values $e(G', G''), G' \neq G''$ are generally small. Inspired by [32], we define the Q-Modularity for the clusters obtained at a hierarchical level i as:

$$Q_i = \sum_{G' \in L_i} \left[e(G', G') - \sum_{G'' \in L_i} e(G', G'') \right]. \tag{11}$$

As regards the classification task, we compare HENPC with four competitor systems which are able to work on data organized in a heterogeneous network. They are:

- **MrSBC** [6][9], which exploits the naïve Bayes classification method in the multi-relational setting.
- **RelIBk** [48], which is the multi-relational version of the k-NN algorithm and is available in RelWeka[10]. As the distance measure, we adopt the Relational Instance-Based Learning (RIBL) measure [21].
- **RelSMO** [48], which is the multi-relational version of Platt's Sequential Minimal Optimization algorithm [35] and is available in RelWeka[10]. This algorithm is based on kernel Support Vector Machines and exploits the kernel Minkowski RIBL set distance [24].
- **GNetMine** [19], a graph-based regularization framework which aims at preserving the network structure of each type of link separately.

[9] `www.di.uniba.it/~ceci/micFiles/systems/MURENA.html`
[10] `kappa.arisco.pl/~adamw/home_page/rel_weka/`

Note that MrSBC, RelIBk and RelSMO perform single-type classification, therefore we learn a model and evaluate the results for each target type in the set $\mathcal{T}_t^*$ separately. Moreover, although GNetMine is, in principle, able to perform multi-type classification, it assumes that all the target types have the same classification scheme (i.e., the set of possible labels is common). We performed some preliminary experiments with GNetMine and discovered that this assumption leads to very poor results when different classification schemes are assigned to different object types (as, for example, in the case of the dataset Yelp). Such a poor result is mainly due to the fact that the system assigns labels belonging to the classification scheme of a target type also to unlabeled nodes of other target types. For this reason, also for GNetMine, in the following we report the results obtained by learning a model for each target type in the set $\mathcal{T}_t^*$ separately.

We evaluate the classification performance according to the accuracy measure, which is defined, for each target type $T_p \in \mathcal{T}_t^*$, as:

$$acc(T_p) = \frac{\sum_{Y \in \mathcal{Y}_p} TP_Y}{|un(V_p)|}, \tag{12}$$

where TP_Y is the number of true positive examples for the class Y.

For both clustering and classification, we average the results obtained with a stratified 10-fold cross validation. For classification, we report the weighted (according to the number of classes) average results obtained on multiple target types.

As regards the parameters of HENPC, we set $c = 3$ (the number of considered shortest meta-paths), in order to guarantee that the best meta-paths are considered, and $depth = 2$ (for the aggregation of attribute values coming from connected nodes which do not appear in any meta-paths). As for the parameter β, since the distribution of scores can vary significantly among datasets, we compute the value of β such that we keep a given percentage of tuples. In particular, we consider 20%, 40% and 60% of tuples. This also allows us to evaluate how the accuracy of the results changes when the number of tuples increases. As regards the parameter α, we follow the recommendations in [33], where a similar merging approach is proposed: We set $\alpha = \beta - 0.2$. Finally, we set γ to the values $\{0.7, 0.8, 0.9\}$, which, according to some preliminary experiments, generally lead to the best results.

The parameters of the competitor systems have been set as follows. HOCCLUS2 requires two parameters, i.e. α and β. Since they have a similar role with respect to α and β of HENPC (i.e., a threshold on the merging procedure and a threshold on the edge weight), we set them to the same values considered for HENPC. MrSBC, RelIBk and RelSMO require the parameter *depth* (*maxPathLength* in MrSBC), which is the maximum depth of analysis from the target nodes. Since its role is similar to that of the parameter *depth* of HENPC, we set it to the same value (i.e., 2). As for the parameter k of RelIBk (i.e., the number of nearest neighbors considered for the classification), we consider the values $\{1, 3, 5\}$. Finally, GNetMine requires a weight for each node type and link type. We leave these weights to the default values, which assume that all node types and link types have the same weight.

4.3. Results

In this section, we describe the results obtained by HENPC and the competitor systems on the considered datasets. First, we analyze the clustering results, which are reported in Table 1.

Overall, we focus on the first three hierarchical levels which appear, according to the clustering and classification results, to be the most significant. We can observe that HENPC is always able to identify at least three hierarchical levels for all the datasets, except for UWCSE. Moreover,

Table 1: Q-Modularity obtained by HOCCLUS2 and HENPC. Bold values indicate the best result obtained for a given dataset and a given percentage of considered instances. L1, L2 and L3 indicate the level of the hierarchy.

DBLP	β: 20%	β: 40%	β: 60%
HENPC L1	**0.64840**	0.60114	**0.54510**
HENPC L2	0.64674	**0.60423**	0.54487
HENPC L3	0.63536	0.60296	0.53310
HOCCLUS2 L1	0.33482	0.07372	0.11779
HOCCLUS2 L2	N/A	N/A	N/A
HOCCLUS2 L3	N/A	N/A	N/A

WEBKB	β: 20%	β: 40%	β: 60%
HENPC L1	**0.98684**	**0.77932**	**0.77937**
HENPC L2	0.97914	0.77868	0.77878
HENPC L3	0.96622	0.77783	0.7773
HOCCLUS2 L1	0.19096	0.19096	0.19096
HOCCLUS2 L2	0.20774	0.20774	0.20774
HOCCLUS2 L3	0.21558	0.21558	0.21558

MOVIES	β: 20%	β: 40%	β: 60%
HENPC L1	**0.90369**	**0.79598**	**0.74514**
HENPC L2	0.90279	0.79357	0.74093
HENPC L3	0.89745	0.78821	0.73239
HOCCLUS2 L1	0.14783	0.10663	0.18794
HOCCLUS2 L2	0.19603	0.1518	0.24356
HOCCLUS2 L3	0.25359	0.20187	0.28913

UWCSE	β: 20%	β: 40%	β: 60%
HENPC L1	**0.99304**	**0.87153**	**0.79293**
HENPC L2	0.98818	0.85076	0.78142
HENPC L3	N/A	N/A	N/A
HOCCLUS2 L1	0.38254	0.3701	0.36977
HOCCLUS2 L2	0.41279	0.40052	0.39647
HOCCLUS2 L3	0.42437	0.41169	0.40862

YELP	β: 20%	β: 40%	β: 60%
HENPC L1	0.76875	**0.77887**	**0.71998**
HENPC L2	**0.80390**	0.76771	0.70514
HENPC L3	0.79622	0.73753	0.69806
HOCCLUS2 L1	0.04046	0.17437	0.09984
HOCCLUS2 L2	0.05029	0.15234	0.07424
HOCCLUS2 L3	N/A	N/A	N/A

IMDB	β: 20%	β: 40%	β: 60%
HENPC L1	**0.82422**	**0.77225**	**0.69099**
HENPC L2	0.79848	0.71135	0.65255
HENPC L3	0.72499	0.66142	0.57746
HOCCLUS2 L1	0.15364	0.4175	0.37215
HOCCLUS2 L2	0.20782	0.47259	0.34343
HOCCLUS2 L3	0.24304	0.39406	0.20905

in some cases, the quality of the clusters obtained increases at the second level of the hierarchy, emphasizing that the merging procedure is able to improve the clustering results. However, it appears that, for the clustering task, the first hierarchical level is generally able to obtain the best results. On the other hand, HOCCLUS2 is not always able to identify at least three hierarchical levels and, in general, the Q-Modularity obtained is significantly lower. This is possibly due to the fact that, in some cases, HOCCLUS2 discards objects that are considered as noise and thus loses possibly significant links. As expected, Q-Modularity is higher for smaller values of β. This is somehow natural since Q-Modularity is higher when we only focus on tuples with a higher strength of the relationship (scores).

In order to statistically compare HENPC with HOCCLUS2, we performed the Friedman test with the Nemenyi post-hoc tests and, following the suggestions made in [10], we plot the graphs which summarize the results in Figure 6. By observing the graphs, we can conclude that HENPC is able to significantly outperform the competitor in all the considered datasets, especially in the case of the first and the second level of the hierarchy. This confirms that, even stopping the algorithm at the first hierarchical level, the behavior of HENPC is robust, independently of the considered dataset.

In order to evaluate whether the clustering quality reflects on the classification accuracy, in Table 2, we report the accuracy values obtained by HENPC on the first three levels of the hierarchy, together with the results obtained by the competitor systems.

As can be observed from the table, HENPC outperforms all the competitors on the datasets UWCSE and WEBKB, independently of the parameters and for almost all the hierarchical levels. In any case, clustering obtained at the first level of the hierarchy guarantees that HENPC outperforms all the competitors in terms of classification accuracy.

As regards MOVIES, we can observe that HENPC, at the first level, obtains a result which

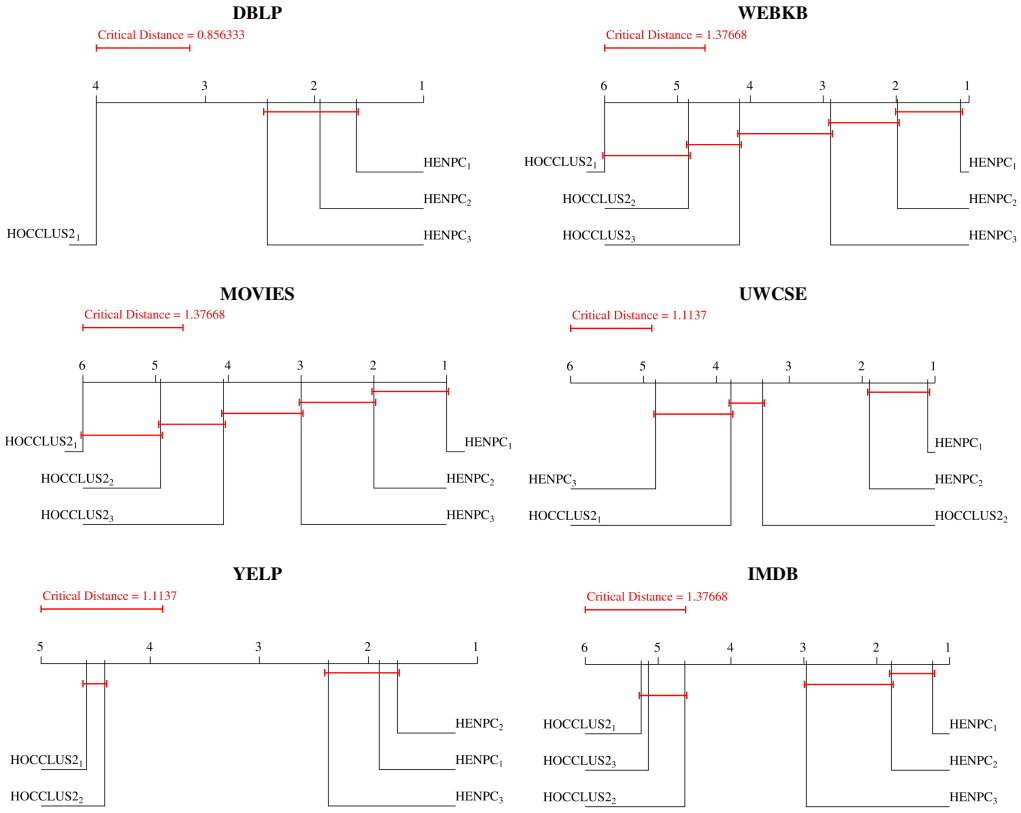

Figure 6: Results of the Nemenyi post-hoc test for the clustering task. Better algorithms are positioned on the right-hand side, and those that do not significantly differ in performance (at p-value=0.05) are connected with a line. The level of the hierarchy is denoted with a subscript in the name of the algorithm.

is comparable to that of RelIBk and RelSMO, and outperforms MrSBC and GNetMine. This is possibly due to the RIBL similarity, which is able to capture similar network characteristics considered by HENPC. On the dataset YELP, HENPC outperforms all the competitors both on single target types and on average, when at least 40% of the tuples are considered (i.e., β such that 40% of tuples is kept), whereas the accuracy degenerates when only 20% of tuples is kept. Moreover, also in this case, the best results are obtained at the first level of the hierarchy. On the dataset DBLP, HENPC outperforms all the competitors at the first level of the hierarchy, when we keep 40% of the tuples. In the other cases, MrSBC is able to obtain better results, probably because of the availability of features whose data distribution is profitably exploited by the naïve Bayes approach. Finally, on the dataset IMDB, HENPC outperforms RelIBk, RelSMO and GNetMine on the first level of the hierarchy, but the best results are obtained by MrSBC, which possibly takes advantages of the availability of the node features.

Table 2: Accuracy obtained by HENPC and the competitors on all the considered datasets. Bold values indicate that HENPC outperforms all the competitors.

System	Params	**UWCSE** T_p: *courses*; S_p: *discipline*			**WEBKB** T_p: *pages*; S_p: *category*			**MOVIES** T_p: *movies*; S_p: *genre*		
RelIBk	$k=1$	0.141			0.606			0.510		
	$k=3$	0.152			0.606			0.510		
	$k=5$	0.152			0.606			0.510		
RelSMO		0.205			0.612			0.510		
MrSBC		0.274			0.510			0.433		
GNetMine		0.253			0.937			0.485		
HENPC	γ	β: 20%	β: 40%	β: 60%	β: 20%	β: 40%	β: 60%	β: 20%	β: 40%	β: 60%
Level 1	0.70	0.254	**0.317**	**0.283**	**0.974**	**0.978**	**0.974**	0.509	0.510	0.510
	0.80	0.254	**0.279**	0.263	**0.974**	**0.978**	**0.974**	0.510	0.510	0.510
	0.90	0.254	**0.290**	0.256	**0.974**	**0.976**	**0.974**	0.510	0.510	0.510
Level 2	0.70	**0.374**	**0.326**	**0.300**	0.804	0.804	0.804	0.390	0.394	0.391
	0.80	**0.374**	**0.336**	**0.300**	0.804	0.804	0.806	0.391	0.394	0.391
	0.90	**0.374**	**0.344**	**0.300**	0.806	0.806	0.808	0.391	0.394	0.391
Level 3	0.70	N/A	**0.312**	**0.283**	0.789	0.790	0.790	0.393	0.397	0.391
	0.80	N/A	**0.304**	**0.283**	0.794	0.792	0.794	0.393	0.397	0.391
	0.90	N/A	**0.330**	**0.283**	0.794	0.798	0.802	0.393	0.397	0.391

System	Params	**YELP** T_p: *business*; S_p: *category*			T_p: *users*; S_p: *average_rate*			μ_t		
RelIBk	$k=1$	0.519			0.560			0.532		
	$k=3$	0.519			0.614			0.551		
	$k=5$	0.516			0.591			0.541		
RelSMO		0.593			0.484			0.557		
MrSBC		0.460			0.600			0.507		
GNetMine		0.497			0.492			0.495		
HENPC	γ	β: 20%	β: 40%	β: 60%	β: 20%	β: 40%	β: 60%	β: 20%	β: 40%	β: 60%
Level 1	0.70	0.377	**0.628**	0.575	0.546	**0.650**	**0.626**	0.433	**0.635**	**0.592**
	0.80	0.372	**0.627**	0.580	0.544	**0.648**	**0.618**	0.429	**0.634**	**0.593**
	0.90	0.422	**0.628**	0.561	0.544	**0.643**	**0.621**	0.463	**0.633**	**0.581**
Level 2	0.70	0.271	0.528	0.520	0.454	**0.617**	0.584	0.332	0.558	0.542
	0.80	0.272	0.536	0.518	0.454	0.614	0.586	0.333	0.562	0.540
	0.90	0.271	0.533	0.490	0.453	0.614	0.586	0.332	0.560	0.522
Level 3	0.70	0.282	0.495	0.484	0.472	**0.625**	0.585	0.345	0.539	0.518
	0.80	0.277	0.506	0.465	0.472	**0.622**	0.580	0.342	0.545	0.503
	0.90	0.277	0.509	0.443	0.472	0.614	0.586	0.342	0.544	0.490

System	Params	**DBLP** T_p: *papers*; S_p: *citation_level*			**IMDB** T_p: *movies*; S_p: *genre*		
RelIBk	$k=1$	0.416			0.501		
	$k=3$	0.433			0.523		
	$k=5$	0.422			0.566		
RelSMO		0.419			0.554		
MrSBC		0.472			0.612		
GNetMine		0.479			0.576		
HENPC	γ	β: 20%	β: 40%	β: 60%	β: 20%	β: 40%	β: 60%
Level 1	0.70	0.442	**0.480**	0.460	0.592	0.592	0.592
	0.80	0.442	0.478	0.460	0.592	0.592	0.592
	0.90	0.447	**0.493**	0.458	0.592	0.592	0.592
Level 2	0.70	0.382	0.428	0.457	0.462	0.454	0.438
	0.80	0.382	0.430	0.455	0.462	0.453	0.438
	0.90	0.383	0.423	0.455	0.462	0.453	0.438
Level 3	0.70	0.390	0.375	0.432	0.463	0.468	0.506
	0.80	0.390	0.373	0.433	0.463	0.468	0.508
	0.90	0.388	0.359	0.435	0.463	0.467	0.509

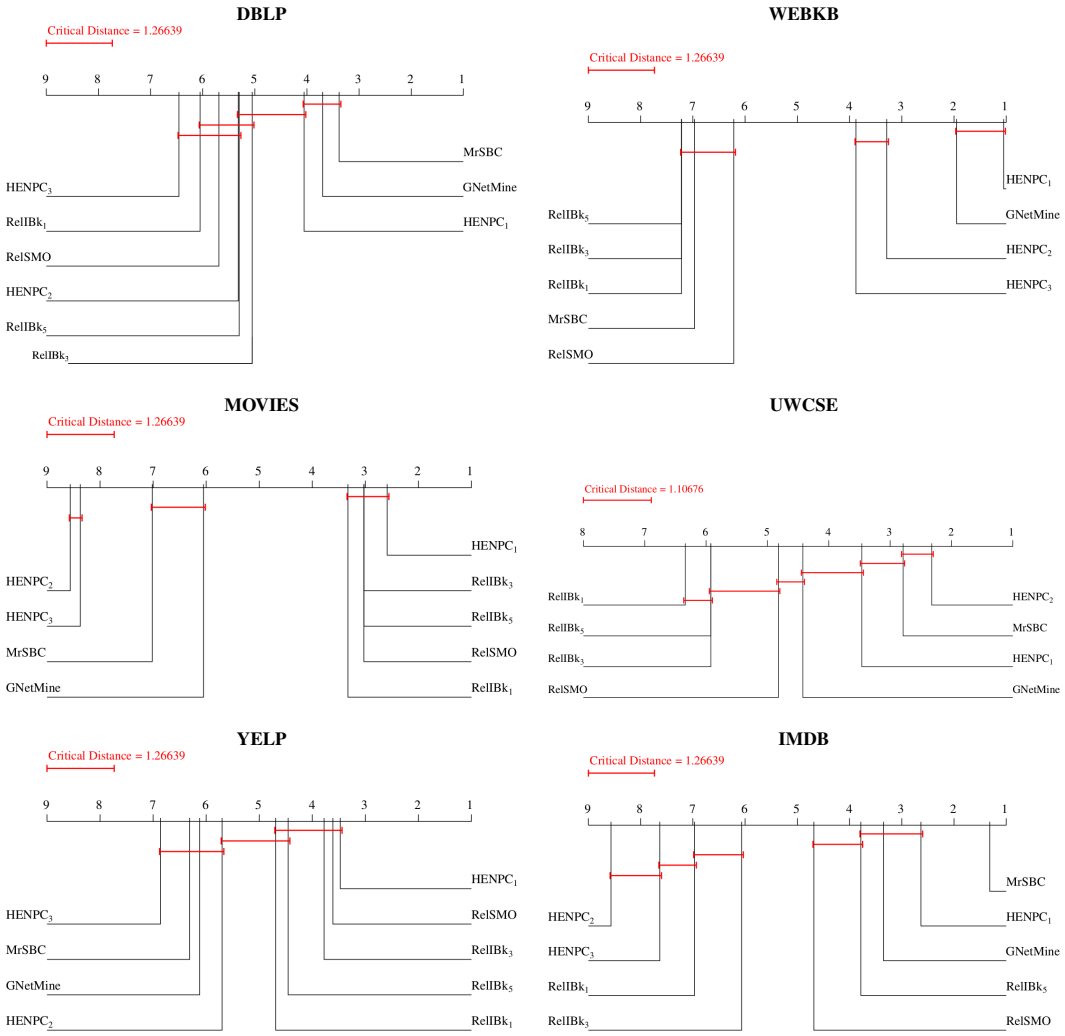

Figure 7: Results of the Nemenyi post-hoc test for the classification task. Better algorithms are positioned on the right-hand side, and those that do not significantly differ in performance (at p-value=0.05) are connected with a line. Subscripts denote the level of the hierarchy.

Table 3: Average running times (in seconds) required by HENPC and competitors on all the considered datasets.

	RelIBk$_1$	**RelIBk$_3$**	**RelIBk$_5$**	**RelSMO**	**MrSBC**	**GNetMine**	**HOCCLUS2**	**HENPC**
DBLP	1.01	0.97	0.96	12.65	4.25	1.45	41.55	2.75
WEBKB	123.12	123.29	123.75	2630.07	2.45	15.57	20.95	92.46
MOVIES	601.47	595.41	605.52	2067.69	13.98	43.50	128.15	441.46
UWCSE	0.16	0.16	0.15	3.88	1.50	0.62	2.12	3.74
YELP	325.94	326.12	327.85	500.61	51.45	262.22	34.36	152.00
IMDB	36.89	37.10	37.43	211.25	31.68	358.71	0.91	1.62

Overall, we can conclude that keeping 40% of tuples, HENPC provides the best classification results. Moreover, as regards the user-defined parameter γ, we can observe that the performance in the range of values examined can be considered comparable. However, by looking at the results in the tables, it is clear that, in general, $\gamma = 0.7$ (which gives a weight of 0.7 to the autocorrelation that relates each target attribute with itself and a weight of 0.3 to the autocorrelation that relates each target attribute with an independent attribute) leads to the best results.

Similarly to the analysis reported for the clustering task, also for the classification task we performed the Friedman test with the Nemenyi post-hoc tests and plotted the graphs which summarize the results in Figure 7. As we can observe from the graphs, the first level of the hierarchy of HENPC leads to the most robust results over the considered datasets. In the case of the DBLP and IMDB datasets, the system MrSBC shows better results, but for DBLP the difference is not statistically significant.

By comparing the clustering and classification results, we can conclude that there is a strong relationship between the clustering quality, measured in terms of Q-Modularity, and the classification quality, measured in terms of accuracy. This result confirms one of the main assumption which guided this work, that is: a strong relationship among a group of heterogeneous objects possibly implies that their class values are correlated.

As a further evaluation, in Table 3 we report the average running time (in seconds) required by HENPC and competitor systems to analyze the considered datasets, on a machine with four 3.5Ghz CPUs and 32GB of RAM. It is noteworthy that the comparison of running times should take into account that HENPC solves both the clustering and the classification tasks, while HOCCLUS2 solves only the clustering task and RelIBk, RelSMO, GNetMine and MrSBC solve only the classification task. From this table, we can observe that HENPC requires a comparable, and often lower, running time with respect to RelIBk, RelSMO and GNetMine. However, if compared with MrSBC, HENPC genrally requires higher running time. This is in line with naïve Bayesian approach implemented in MrSBC. However, in the three cases where MrSBC is faster than HENPC (WEBKB, MOVIES and YELP) the accuracy obtained by HENPC is significantly higher than that obtained by MrSBC (see Figure 7).

5. Conclusion

In this work, we consider the task of multi-type clustering and classification in heterogeneous networks, i.e. with multiple types of nodes and edges, organized in an arbitrary structure. In order to perform this task, we propose the algorithm HENPC, which extracts hierarchically organized, possibly overlapping and heterogeneous clusters, and exploits them for predictive purposes. The proposed algorithm is four-stepped: *i)* the identification of the strength of the relationships among target nodes, which exploits the whole network structure; *ii)* the identification of an initial set of heterogeneous clusters in the form of multi-type cliques; *iii)* the construction of a hierarchy of

heterogeneous clusters, which allows the algorithm to possibly catch autocorrelation phenomena at different levels of granularity; *iv)* the identification of classification functions by exploiting the identified clusters.

The experiments performed on real datasets show that HENPC is able to outperform competitors in terms of both clustering quality and classification accuracy. Moreover, we observed the presence of a relationship between the clustering quality, measured in terms of Q-Modularity, and the classification quality, measured in terms of accuracy. This confirms the assumption that the identification of groups of strongly connected objects, even if they are heterogeneous, can be exploited to catch autocorrelation phenomena and possibly improve the classification accuracy.

For future work, we intend to investigate the possibility of extending HENPC to the *inductive* semi-supervised setting (not only *transductive*). Moreover, we will consider the exploitation of the discovered heterogeneous clusters for the prediction of previously unknown links between objects (heterogeneous link prediction).

Acknowledgments

We would like to acknowledge the support of the EU Commission through the project MAESTRA - Learning from Massive, Incompletely annotated, and Structured Data (Grant number ICT-2013-612944). The authors wish to thank Lynn Rudd for her help in reading the manuscript.

References

[1] Angin, P., Neville, J., 2008. A shrinkage approach for modeling non-stationary relational autocorrelation, in: Proc. IEEE ICDM, IEEE Computer Society. pp. 707–712.

[2] Appice, A., Ceci, M., Malerba, D., 2009. An iterative learning algorithm for within-network regression in the transductive setting, in: Proc. of Discovery Science, Springer. pp. 36–50.

[3] Barracchia, E.P., Pio, G., Malerba, D., Ceci, M., 2017. Identifying lncRNA-disease Relationships via Heterogeneous Clustering, in: New Frontiers in Mining Complex Patterns - 6th International Workshop, NFMCP 2017, Held in Conjunction with ECML-PKDD 2017, Skopje (Macedonia).

[4] Bilgic, M., Getoor, L., 2008. Effective label acquisition for collective classification, in: Proc. 14th ACM SIGKDD Intl. Conf on Knowledge Discovery and Data Mining, ACM. pp. 43–51.

[5] Blockeel, H., De Raedt, L., Ramon, J., 1998. Top-down induction of clustering trees, in: Proc. 15th Intl. Conf. on Machine Learning, Morgan Kaufmann. pp. 55–63.

[6] Ceci, M., Appice, A., Malerba, D., 2003. Mr-SBC: a multi-relational naive bayes classifier, in: Knowledge Discovery in Databases: PKDD 2003. Springer, pp. 95–106.

[7] Ceci, M., Cuzzocrea, A., Malerba, D., 2015a. Effectively and efficiently supporting roll-up and drill-down OLAP operations over continuous dimensions via hierarchical clustering. J. Intell. Inf. Syst. 44, 309–333.

[8] Ceci, M., Pio, G., Kuzmanovski, V., Dzeroski, S., 2015b. Semi-supervised multi-view learning for gene network reconstruction. PLOS ONE 10, 1–27.

[9] Chuhay, R., 2010. Marketing via Friends: Strategic Diffusion of Information in Social Networks with Homophily. Technical Report 2010.118. Fondazione Eni Enrico Mattei.

[10] Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30.

[11] Desrosiers, C., Karypis, G., 2009. Within-network classification using local structure similarity, in: Proc. of ECML PKDD 2009, Springer-Verlag, Berlin, Heidelberg. pp. 260–275.

[12] Gallagher, B., Tong, H., Eliassi-Rad, T., Faloutsos, C., 2008. Using ghost edges for classification in sparsely labeled networks, in: Proc. ACM SIGKDD, ACM. pp. 256–264.

[13] Gao, B., Liu, T.Y., Zheng, X., Cheng, Q.S., Ma, W.Y., 2005. Consistent bipartite graph co-partitioning for star-structured high-order heterogeneous data co-clustering, in: Proc. ACM SIGKDD, ACM, New York, NY, USA. pp. 41–50.

[14] Grcar, M., Trdin, N., Lavrac, N., 2013. A methodology for mining document-enriched heterogeneous information networks. Comput. J. 56, 321–335.

[15] Han, J., Kamber, M., Pei, J., 2006. Data Mining: Concepts and Techniques, Second Edition (The Morgan Kaufmann Series in Data Management Systems). 2 ed., Morgan Kaufmann.

[16] Ienco, D., Robardet, C., Pensa, R., Meo, R., 2013. Parameter-less co-clustering for star-structured heterogeneous data. Data Mining and Knowledge Discovery 26, 217–254.
[17] Jensen, D., Neville, J., Gallagher, B., 2004. Why collective inference improves relational classification, in: Proc. ACM SIGKDD, ACM. pp. 593–598.
[18] Ji, M., Han, J., Danilevsky, M., 2011. Ranking-based classification of heterogeneous information networks, in: Proc ACM SIGKDD, ACM, New York, NY, USA. pp. 1298–1306.
[19] Ji, M., Sun, Y., Danilevsky, M., Han, J., Gao, J., 2010. Graph regularized transductive classification on heterogeneous information networks, in: Machine Learning and Knowledge Discovery in Databases. Springer Berlin Heidelberg. volume 6321 of *LNCS*, pp. 570–586.
[20] Kirsten, M., Wrabel, S., Horváth, T., 2000. Distance based approaches to relational learning and clustering, in: Dězeroski, S. (Ed.), Relational Data Mining. Springer-Verlag New York, Inc., New York, NY, USA, pp. 213–230.
[21] Kirsten, M., Wrobel, S., Horvath, T., 2001. Distance based approaches to relational learning and clustering, in: Relational data mining. Springer, pp. 213–232.
[22] Kong, X., Cao, B., Yu, P.S., Ding, Y., Wild, D.J., 2013. Meta path-based collective classification in heterogeneous information networks. CoRR abs/1305.4433.
[23] Kong, X., Shi, X., Yu, P.S., 2011. Multi-label collective classification, in: Proc. of SIAM SDM 2011, SIAM / Omnipress. pp. 618–629.
[24] Kruskal, J.B., 1964. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. Psychometrika 29, 1–27.
[25] Li, T., Anand, S.S., 2008. Hirel: An incremental clustering algorithm for relational datasets, in: ICDM, IEEE Computer Society. pp. 887–892.
[26] Long, B., Zhang, Z.M., Wú, X., Yu, P.S., 2006. Spectral clustering for multi-type relational data, in: Proceedings of the 23rd international conference on Machine learning, ACM, New York, NY, USA. pp. 585–592.
[27] Lu, Q., Getoor, L., 2003. Link-based classification, in: Fawcett, T., Mishra, N. (Eds.), Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA, AAAI Press. pp. 496–503.
[28] Macskassy, S.A., Provost, F., 2007. Classification in networked data: A toolkit and a univariate case study. J. Mach. Learn. Res. 8, 935–983.
[29] McPherson, M., Smith-Lovin, L., Cook, J., 2001. Birds of a feather: Homophily in social networks. Annual Review of Sociology 27, 415–444.
[30] Neville, J., Adler, M., Jensen, D.D., 2003. Clustering relational data using attribute and link information, in: Proceedings of the Workshop on Text Mining and Link Analysis, Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico.
[31] Newman, M., Barabasi, A.L., Watts, D.J., 2006. The Structure and Dynamics of Networks: (Princeton Studies in Complexity). Princeton University Press, Princeton, NJ, USA.
[32] Newman, M.E.J., 2006. Modularity and community structure in networks. Proceedings of the National Academy of Sciences 103, 8577–8582.
[33] Pio, G., Ceci, M., D'Elia, D., Loglisci, C., Malerba, D., 2013. A Novel Biclustering Algorithm for the Discovery of Meaningful Biological Correlations between microRNAs and their Target Genes. BMC Bioinformatics 14, S8.
[34] Pio, G., Ceci, M., Malerba, D., D'Elia, D., 2015. ComiRNet: a web-based system for the analysis of miRNA-gene regulatory networks. BMC Bioinformatics 16, S7.
[35] Platt, J., 1998. Fast training of support vector machines using sequential minimal optimization, in: Advances in Kernel Methods - Support Vector Learning. MIT Press.
[36] Raedt, L.D., Lavrac, N., 1996. Multiple predicate learning in two inductive logic programming settings. Logic Journal of the IGPL 4, 227–254.
[37] Rahmani, H., Blockeel, H., Bender, A., 2010. Predicting the functions of proteins in protein-protein interaction networks from global information. Journal of Machine Learning Research 8, 82–97.
[38] Saha, T., Rangwala, H., Domeniconi, C., 2012. Multi-label collective classification using adaptive neighborhoods, in: Prof of ICMLA 2012. Volume 1, IEEE. pp. 427–432.
[39] Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T., 2008. Collective classification in network data. AI Magazine 29:3, 93–106.
[40] Steinhaeuser, K., Chawla, N.V., Ganguly, A.R., 2011. Complex networks as a unified framework for descriptive analysis and predictive modeling in climate science. Statistical Analysis and Data Mining 4, 497–511.
[41] Stojanova, D., Ceci, M., Appice, A., Dzeroski, S., 2012. Network regression with predictive clustering trees. Data Min. Knowl. Discov. 25, 378–413.
[42] Stojanova, D., Ceci, M., Malerba, D., Dzeroski, S., 2013. Using PPI network autocorrelation in hierarchical multi-label classification trees for gene function prediction. BMC Bioinformatics 14, 285.
[43] Sun, Y., Aggarwal, C.C., Han, J., 2012. Relation strength-aware clustering of heterogeneous information networks with incomplete attributes. Proc. VLDB Endow. 5, 394–405.

[44] Sun, Y., Han, J., Zhao, P., Yin, Z., Cheng, H., Wu, T., 2009a. RankClus: integrating clustering with ranking for heterogeneous information network analysis, in: Proc of EDBT 2009, ACM, New York, NY, USA. pp. 565–576.

[45] Sun, Y., Yu, Y., Han, J., 2009b. Ranking-based clustering of heterogeneous information networks with star network schema, in: Proc of SIGKDD 2009, ACM, New York. pp. 797–806.

[46] Vandic, D., van Dam, J.W., Frasincar, F., 2012. A semantic-based approach for searching and browsing tag spaces. Decision Support Systems 54, 644 – 654.

[47] Wang, J., Zeng, H., Chen, Z., Lu, H., Tao, L., Ma, W.Y., 2003. ReCoM: reinforcement clustering of multi-type interrelated data objects, in: Proc of SIGIR 2003, ACM, New York. pp. 274–281.

[48] Woznica, A., Kalousis, A., Hilario, M., 2007. Learning to combine distances for complex representations, in: Machine Learning, Proc ICML 2007, pp. 1031–1038.

[49] Yin, X., Han, J., Yu, P.S., 2006. LinkClus: efficient clustering via heterogeneous semantic links, in: Proc of VLDB 2006, VLDB Endowment. pp. 427–438.

[50] Yin, X., Han, J., Yu, P.S., 2007. Crossclus: user-guided multi-relational clustering. Data Min. Knowl. Discov. 15, 321–348.

[51] Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, P.B., 2004. Learning with local and global consistency, in: Thrun, S., Saul, L.K., Schölkopf, P.B. (Eds.), Advances in Neural Information Processing Systems 16. MIT Press, pp. 321–328.

[52] Zhu, X., Ghahramani, Z., Lafferty, J.D., 2003. Semi-supervised learning using gaussian fields and harmonic functions, in: Proc. 20th Intl. Conf. on Machine Learning, AAAI Press. pp. 912–919.

Appendix - Summary of the symbols used in the paper

Symbol	Description
$G = (V, E)$	Heterogeneous information network, represented as a graph
V	Set of nodes of the graph G
E	Set of edges of the graph G
$G' = (V', E')$	Generic heterogeneous cluster
V'	Set of nodes of the heterogeneous cluster G'
$E' \subseteq (E \cup \hat{E})$	Set of edges of the heterogeneous cluster G'
$\hat{E}$	Set of edges identified by the clustering method
$v', v'' \in V$	Generic nodes
$t_v(v')$	Node type of node v'
$\mathcal{T} = \mathcal{T}_t \cup \mathcal{T}_{tr}$	Set of possible node types
$\mathcal{T}_t$	Target types
$\mathcal{T}_t^*$	Target types with an associated classification scheme
$\mathcal{T}_{tr}$	Task-relevant types
T_p, T_q	Generic node types
$V_p \subseteq V$	Subset of nodes of type T_p
$\mathcal{X}_p$	Set of attributes $\{X_{p,1}, \ldots, X_{p,m_p}\}$ associated to nodes of type T_p
G'_p	Set of nodes of type T_p in G'
$lab(G'_p)$	Set of labeled nodes of type T_p in G'
$un(G'_p)$	Set of unlabeled nodes of type T_p in G'
$\mathcal{R}$	Set of possible edge types
R_j	Generic edge type
$e = \langle R_j, \langle v', v'' \rangle \rangle$	Generic edge of type R_j between nodes v' and v''
$t_e(e) = R_j$	Type of edge $e = \langle R_j, \langle v', v'' \rangle \rangle$
E_j	Subset of edges of type R_j
L_i	Generic hierarchy level (a set of heterogeneous clusters)
k	Number of hierarchy levels
$\mathcal{S}$	Set of possible classification schemes
S_p	Classification scheme associated to nodes of type T_p
$\phi : \mathcal{T}_t \rightarrow \mathcal{S}$	Function which maps a target type to its classification scheme
$\psi_p^{(i)} : V_p \rightarrow \mathcal{Y}_p$	Classification function for nodes of type T_p, based on level L_i
$\mathcal{Y}_p$	Range of the functions $\{\psi_p^{(i)}\}_{i=1,\ldots k}$, defined according to scheme S_p
$Y \in \mathcal{Y}_p$	Generic value of the function range $\mathcal{Y}_p$
$cl(v)$	A possible class label to associate to v
$\vec{v_i}$	Generic tuple of target nodes
s_i	Strength of the relationship among all the nodes in tuple $\vec{v_i}$
$v_{i,j}$	j-th node of tuple v_i
P	Generic meta-path connecting two target nodes
seq_d	Sequence of nodes
$X_C(seq_{d'}, seq_{d''})$	Target attributes of the nodes of the two sequences $seq_{d'}$ and $seq_{d''}$
$X_{NC}(seq_{d'}, seq_{d''})$	Non-target attributes of the nodes of the two sequences $seq_{d'}$ and $seq_{d''}$
$s_x(seq_{d'}, seq_{d''})$	Similarity between the values of attribute x in $seq_{d'}$ and $seq_{d''}$
$val_x(seq_{d'})$	Value of attribute x in a generic sequence $seq_{d'}$
$h(G''')$	Heuristic function which evaluates the quality of cluster G'''
α	Parameter which controls the quality of the hierarchical clustering
β	Threshold on the strength of the relationships among objects
γ	Weight on the autocorrelation that relates the target attribute with itself