Validating Vector-Label Propagation for Graph Embedding

Valerio Bellandi[1], Ernesto Damiani[1], Valerio Ghirimoldi[1], Samira Maghool[1(✉)], and Fedra Negri[2]

[1] Università degli Studi di Milano (UNIMI), Milan, Italy
{valerio.bellandi,ernesto.damiani,samira.maghool}@unimi.it,
valerio.ghirimoldi@studenti.unimi.it
[2] Università degli Studi di Milano-Bicocca, Milan, Italy
fedra.negri@unimib.it

Abstract. Structural network analysis retrieves the holistic patterns of interactions among network instances. Due to the unprecedented growth of data availability, it is time to take advantage of Machine Learning to integrate the outcome of the structural analysis with better predictions on the upcoming states of large networks. Concerning the existing challenges of adopting methods embracing *multi-dimensional*, *multi-task*, *transparent* representations within *incremental* procedures, in our recent study, we proposed the AVPRA algorithm. It works as an embedder of both the network structure and domain-specific features making the aforementioned challenges feasible to address. In this paper, we elaborate on the validation of AVPRA by adopting it in multiple downstream Machine Learning tasks on the Twitter network of the Italian Parliament. Comparing the outcome with state-of-the-art algorithms of graph embedding, the capability of AVPRA in retaining either network structure properties or domain-specific features of the nodes is promising. In addition, the method is incremental and transparent.

Keywords: Vector-label propagation · Social network analysis · Graph embedding

1 Introduction

Network analysis comprises powerful methods to study the relationship between the elements of a network, with applications to any domain where the structure of the network can reveal interaction patterns or emerging states [10]. Most of the applied algorithms exploit graph theory to measure specific properties of the overall network, the subgraphs composing it, or the individual nodes. These measures have been successfully applied to predict the evolution of specific states of the network or its individual nodes [8]. Integrating the results of network analysis into downstream Machine Learning (ML) has become an interesting research topic due to the unprecedented growth of data availability. Leveraging ML algorithms for studying complex networks, containing large number of nodes

M. Sellami et al. (Eds.): CoopIS 2022, LNCS 13591, pp. 259–276, 2022.
https://doi.org/10.1007/978-3-031-17834-4_15

and edges with various properties, have made a huge step toward analysing and predicting the behavior of a system [34]. However, specific challenges have to be addressed to realize this integration.

The metrics from network analysis can be included in the feature space assigned to a learning procedure as any other descriptor, accounting for the structural properties of the elements of the system. Data fusion techniques offer multiple strategies to unify them with descriptors from other sources [21,25]. Nevertheless this approach suffers from the *structural determinism* of network measures i.e. results are determined by the network structure, only disregarding the role of others features characterizing individual elements in the domain [11]. For example, nodes from different communities can be represented by similar centrality values if they have similar structural relationships within the community. Thus, they may fall close in the feature space feeding a ML algorithm, even if belonging to different communities. An alternative strategy is using graph embedding algorithms to represent the network structure in low dimensional space while preserving the distances between the elements in the network [33]. These approaches generate a latent space losing a *transparent* connection to the original network space, thus, even if the distances between nodes are preserved, it is hard to explain the reasons motivating an interspace between elements. Another strategy is using graph neural networks to directly encode the network structure into the architecture of the neural network [1]. This method implies the neural network is designed to address a *specific task*. Updates in the network structure require re-initialising the neural network and the results obtained can be hardly *incrementally* integrated with results obtained in the past. Exploring a system through networks requires tools reflecting the networks features in an interpretable manner. Also, the nodes of a network often refer to dynamically changing instances. Analysing and predicting the dynamics of a network demand to measure its structural properties as well as the full picture of domain-specific features of nodes. In other words, the current methods are *structural determined* and *non transparent*, *task specific* and *non incremental*.

In our previous paper, addressing this matter, we proposed the agent-based Vector-label PRopagation Algorithm (AVPRA) [37] for explainable exploration through the network's feature space. Using this algorithm results in extracting the *d-dimensional* weighted vectors of features in the feature space rather than latent space, which makes them explainable. Each element of these vectors conveys information on how features formed the current status of instances in the network. In this work, we aim at verifying the applicability of AVPRA to different predictive tasks using established ML methods. More specifically, in Sect. 2 we discuss the related works, in Sect. 3 we present the AVPRA algorithm and characterize its properties, in Sect. 4 we present the experimental validation we conducted, in Sect. 5 we discuss the results achieved, and in Sect. 6 we go to the conclusion.

2 Related Works

2.1 Vector-Label Propagation

The original idea of Label Propagation was proposed in 2002, as an iterative procedure to augment data before feeding a classification task [42]. Applying this algorithm, the unlabeled nodes of the network adopt the label of the majority of nodes in their neighborhood. Considering that labels are deterministic and discrete representations of nodes' features by a single value, they are not able to picture a holistic view of the network using information from different sources. Therefore, the Vector-Label Propagation (VLP) algorithms are proposed to encode the multiple features a node could convey in a dynamic system into a vector of labels. In these algorithms, the most weighted labels or combination of all labels of the vector-labels (VL) are utilized to get the most influential features for the given node or to find out the overlapping communities it belongs to [14]. For example, to identify a list of songs to recommend to a user based on the most favored in its neighborhood.

The belonging coefficient of a label is a parameterized measurement computed aggregating the vector-labels in the neighborhood to take account of the influence a label has on the node based on how frequently it appears in the neighborhood. The aggregation procedure acquiring the neighborhood features, a.k.a *update rule*, does not necessarily follow the majority rule. A common complexity may arise when multiple labels have equal frequency hence a *random selection* should be operated. This random choice has been identified as a major source of instability, since different executions of the algorithm may result in different label assignments. To overcome this issue multiple variants of the update rule have been proposed [18,20,35,39–41], using the structural feature of nodes to reduce this type of instability. Primarily, these variants identify the most influential nodes to decide the order in which nodes are updated or the initialization of the labels in the network [2]. Overall, the VL approach helps in reducing the instability of LP algorithms as all the labels in the neighborhood can be accounted for. It also permits considering non-structural features to describe node properties in a holistic multidimensional mode.

2.2 Graph Embedding

Graph Embedding (GE) in general terms is a data preparation procedure mapping the existing information of a network as much as possible in a unified usable data format to input ML algorithms for implementing several downstream tasks. Many algorithms such as Matrix factorization-based [9,27], Random walk-based [15,29,30], and Neural network-based algorithms [16,19] are developed in order to retrieve as much as possible the properties of the network and map them in a latent space [3]. Most of these algorithms have been however criticized for adding a level of complexity to ML results [5]. The latent space they provide implies explainability issues, therefore some post-hoc interpretative analysis is needed [28].

By AVPRA, we propose to adopt a VLP algorithm as a mapping function where the belonging coefficients capture the *diffusibility* of the labels into nodes, depending on the distances from the source of the features and their frequency. To encode the diffusibility of all the features in the network, the *vector-label* includes all of them in vectors of fixed length, while the *belonging coefficient* accounts for their incidence in the node neighborhood. A big advantage of this approach is that, at the end of the propagation procedure, all the nodes of the network can be positioned in the same feature space and the output data format is suitable for implementing multiple ML tasks.

3 Methodology

We first, briefly, introduce the AVPRA algorithm characterizing the procedures it activates in exploring the space of a network and in encoding the collected information into VL.

3.1 AVPRA Model Specification

Considering the rationale behind VLP algorithms in preserving information on the features of individual nodes, we proposed an agent-based model where nodes have memory about received information from different sources and limited rationality for updating its vector label coefficients [37].

The updating rule is realized by an aggregation function unifying the received information from neighborhood. An example of such function is represented in Eq. 1. At each time step t, $b(l)$, the belonging coefficient of an element of the $\mathbf{VL}_i[l](t)$, can be updated by aggregating the k neighbors' $\mathbf{VL}_{j\in\Gamma(i)}[l](t-1)$.

$$\mathbf{VL}_i[l](t) = w_1\mathbf{VL}_i[l](t) + w_2 \sum_{j\in\Gamma(i)} \mathbf{VL}_{j\in\Gamma(i)}[l](t-1) \tag{1}$$

where w_1 and w_2 are the weight of current assigned labels l of node i and the weight of the neighbors $\Gamma(i)$, respectively. In a basic scenario, $w_1 = w_2 = \frac{1}{\Gamma(i)+1} = \frac{1}{k+1}$, hence, for all the common elements in $\mathbf{VL}_i$ and $\mathbf{VL}_j$ vectors, the values of the given elements l increase unconditionally and will be normalized to 1 by the inverse of the cardinality of $\Gamma(i)$. All the **VL**s get updated synchronously, to avoid conflicts in the order of update, and reflect all the changes received at the same iteration by all nodes. Following the dynamical changes of VLs, we witnessed the propagation procedure to reach the termination point, where the updated value of each element is below a certain threshold defined by the user, after a few number of iterations which is about the average path length of the network [37].

Our algorithm mechanism helps in discarding couple of instabilities caused by *random selection* and *ordering* of updates. The former is addressed by the fact that we keep all the coefficient labels up to the termination point, while the latter is addressed by averaging over a number of executions starting with different initial seeds, i.e. the nodes settled as starting points of the propagation process.

As an illustrative example of our proposal, Fig. 1 proposes a schematic network containing three agents with three initial distinct features red (R), yellow (Y) and green (G). The vector-label for agent i at time $t-1$ is as $\{R, Y, G\}_i = \{l_1, l_2, l_3\}_i$, containing the weights of all unique features. The updated vector-label at time t is created as a combination of its neighbors' vector-labels at time $t-1$, according to the update rule.

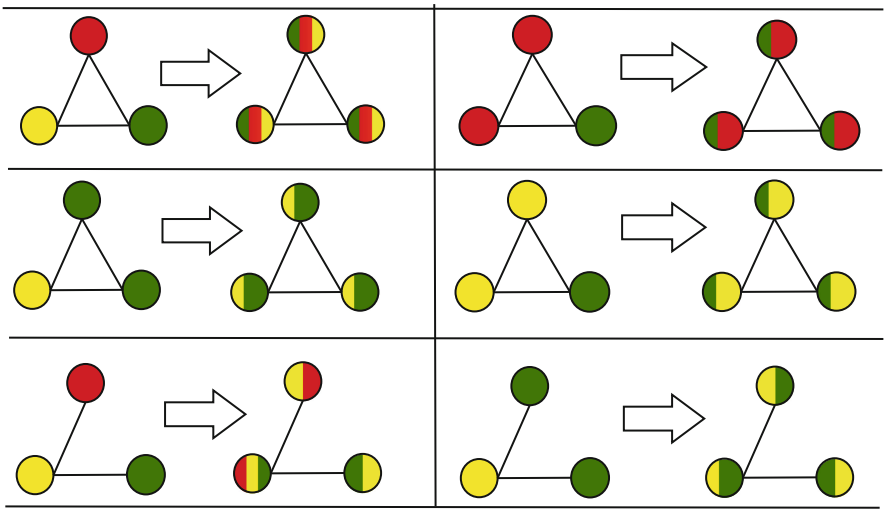


Fig. 1. The schematic view of the AVPRA algorithm implemented on a small network containing three nodes. In the first and second row, the network is fully connected and the states of nodes are depicted by colors; Red, Green, and Yellow. In each section of the picture, the initial state at time $T = t - 1$ is followed by evolved state at $T = t$ according to a propagation rule. In the third row, the connections have changed and subsequently the following state has been modified. (Color figure online)

3.2 Properties of the AVPRA Algorithm

The AVPRA algorithm is an agent-based iterative procedure where agents progress the propagation acting in response to the VLs they access in their neighborhood. To avoid conflicts in propagating the updates, all agents get updated simultaneously at each iteration. The updating rule defines the aggregation function to apply in updating VLs. To initialize the procedure, some of the agents of the network must be assigned to valued VLs but not all of them have to start with an initial valued VL. The termination of the AVPRA algorithm is achieved when the system reaches an iteration s where VLs are stationary. This implies, that the variations on the belonging coefficient of all labels must be less than a defined value p. We name p the negligibly threshold. Experimental results have shown this threshold can be achieved with a number of iterations that is close to the average path length of the network [37].

These properties of the algorithm make it *incremental*, i.e., updates in the network structure do not require re-execute the procedure but can be applied first locally and then propagated in the other nodes by a few numbers of iterations. This makes the AVPRA algorithm more ductile to integrate into downstream ML pipelines.

3.3 Validating the Representing Model

To verify the quality of the representation achieved by the vectors obtained using AVPRA we exploit a comparative approach. The idea is that high-quality graph embeddings should be able to capture key parts of the network structure [7], thus can predict common measures of the network, current or future states. We then compare the accuracy achieved by supervised and unsupervised ML algorithms in predicting these measures and states encoding network nodes using AVPRA VLs or using other state-of-the-art graph embedding methods. Our tests aim to address different relevant *analytical tasks* in network analysis, namely community and sub-community detection (Sect. 4.1 and 5.2) clustering (Sect. 4.2) link prediction (Sect. 4.3) measuring node centrality (Sect. 4.4) and label-drift detection (Sect. 4.5).

In addition to an assessment of the accuracy, our experiments underline capabilities provided by the AVPRA VLs not supported by other embedding methods. Encoding nodes into a feature space that is provided by the union of the features of the nodes in the network, AVPRA keeps *domain properties* transparent to data analysis. To let emerge these aspects, instead of adopting a standard dataset, used in research settings, we preferred to refer to a real-world dataset, where domain-related aspects can emerge.

3.4 Populite Data Set

In a collaborative study funded by the University of Milan, involving the Department of Social and Political Sciences and the Department of Computer Science, the *Populite* project has been launched. The aim is to study the behavioral patterns of Italian politicians on social media. A key aspect of this study is to depict the communities and sub-communities that the communication flow and the social network among Italian politicians on social media create. By studying the inter and intra-cohesion of these communities multiple interesting questions can be answered. Which are the political groups that interact the most, which ones are partitioned into sub-communities (i.e., intra-party factions), and to which other political groups these sub-communities are connected? Individual links can also be analyzed. Is there reciprocity between parliamentarians? Which ones are similar, based on their "neighborhood"?

Considering the Twitter social network, its API can be used to retrieve and analyze the social network and the communication flow of a set of users. This API provides the access to different resources including: *Tweets*, *Users*, *Lists*, *Trends*, *Media* and *Places*. The Political Science Department provided the list of parliamentarians, deputies and senators, and other relevant political actors in Italy. With this information in hand 41615 tweets were downloaded, in the period between January 2020 and February 2022. Information about the membership of the analyzed actors in political groups was retrieved using the open data of the Italian chamber and senate and adjusted manually for those active actors not titled of a seat in the parliament. Data were stored in a graph database to aggregate them and execute the required queries. For example, the social network of the politicians was created by using the *friendship* relation among Twitter users and extracted as the edge list matrix, to input network analysis algorithms.

4 Experimental Results

We proceed with evaluating the AVPRA outcome in implementing some downstream ML tasks exploring the Populite data set. For each task, the accuracy and execution run time are compared with some state-of-the-art algorithms, namely, Deepwalk [29], Node2vec [15], and M-NMF[38].

4.1 Community Detection

In network analysis, communities reveal the structural and functional properties of groups of nodes. Even though there is valuable literature on community detection algorithms, they mostly focus on the structural properties of the networks. In the other words, the fact that each member may be characterized by various domain-specific features toward different communities is not exploited.

In traditional approaches, the community detection task is based on the modularity maximization problem. Modularity measures the strength of division of a network into modules, i.e. strongly connected components. To address this problem, some repetitive steps take place in forming/deforming the possible partitions in the network and finding the one which results in the maximum value of modularity [6]. Among the common algorithms, the so-called *Leiden* community detection proves that the connected communities and all subsets of all partitions are locally optimally assigned [36].

Considering the Twitter connections[1] of the Italian parliament as a directed network, we take the detected communities by Leiden algorithm as ground truth labels. Leveraging these labels we run a Random Forest (RF) classifier in order to evaluate how the vectors resulting from running AVPRA or other graph embedding methods can capture the structural properties of the network similarly to the Leiden algorithm. Table 1 presents the F1 scores and the execution time for

[1] In particular we used the friendship relation accessible from Twitter API.

this task, comparing the AVPRA vector labels with other algorithms embedded vectors. AVPRA MW and AVPRA 10 MW are two derivatives of the AVPRA vector labels where a limited number of labels is taken into account. The former refers to the most weighted label, the latter to the ten most weighted labels. In the case of multi-class classification, some averaging metrics for F1 scores are used in the classification report. A macro-average will compute the metric independently for each class and then take the average hence treating all classes equally, whereas a micro-average will aggregate the contributions of all classes to compute the average metric. In the case of an equal number of samples for each class, macro and micro averaging will result in the same score [26]. For those algorithms with hyperparameters in the mapping function, the mean and standard deviation of F1 scores are calculated.

Table 1. The evaluation of AVPRA used for the community detection task. A RF classifier is trained on the 80% of VLs with the Leiden communities labels and tested on the 20% calculating the F1 score.

	Community detection ($mean \pm std$)		
Embedding algorithms	F1_micro	F1_macro	Execution_time(s)
AVPRA	**0.993464**	**0.993073**	3.768
AVPRA MW	0.967320	0.966128	1.879
AVPRA 10MW	0.967320	0.967105	0.627
DeepWalk	0.975163 ± 0.01164	0.97322 ± 0.01219	3.545
Node2vec	0.972222 ± 0.0119	0.971005 ± 0.1198	11.681
M-NMF	0.972549 ± 0.01503	0.971930 ± 0.0146	39.003/18.921

4.2 Clustering

Continuing the evaluation of the representation power of the AVPRA VLs, an unsupervised task is studied. In particular, we consider the capabilities of clustering techniques in grouping similar instances. A number of clusters equal to the number of partitions proposed by the Leiden community detection algorithm is formed out of the VLs by AVPRA and, embedded vectors by Deepwalk, Node2vec, and M-NMF. In the evaluation of unsupervised tasks, Normalized Mutual Information (NMI) is a measure used to evaluate network partitioning performed by community detection algorithms. The basic idea of this metric represents the amount of retrieved information from one distribution regarding the second one [23]. Spectral, Kmeans, and Agglomerative Clustering algorithms are applied to the Populite network, and the calculated NMI for each of these algorithms on embedding methods is presented in Table 2.

Table 2. The Normalized Mutual Information, comparing the clustered embedded vectors resulted by one clustering algorithm.

	NMI (*mean* ± *std*)		
Algorithm	Spectral clustering	Kmeans Clustering	Agglomerative Clustering
AVPRA	0.892796	0.883197	**0.882575**
DeepWalk	0.903192 ± 0.043	0.926282 ± 0.033	0.867466 ± 0.028
Node2vec	**0.926852±0.009**	**0.932014±0.011**	0.872850 ± 0.012
M-NMF	0.86433 ± 0.055	0.867234 ± 0.048	0.845866 ± 0.044

4.3 Link Prediction

The link prediction problem mainly refers to the evaluation of possible relations between two existing nodes in the network [17,22]. Often the problem is addressed using a supervised learning approach, where a model is trained based on the existing/corrupted links in the network [31]. In our experiments, we adopted a direct evaluation approach. After getting the vectors by the compared embedding algorithms, for each possible couple of nodes in the network (each possible edge), we calculate the cosine similarity of their assigned vectors in the mapping space. This way, for each node based on the maximum similarity with other nodes, we predict the presence of the edge if the similarity value is more than 0.5. After all, we evaluate the presence or absence of a predicted edge based on the true edges in the reference network. Table 3 illustrates the results achieved.

Table 3. The comparison of link prediction score using different embedding algorithms.

	Link prediction (*mean* ± *std*)	
Embedding algorithms	F1_micro	Execution_time(s)
AVPRA	**0.77790**	0.627
DeepWalk	0.745352 ± 0.0252	13.178
Node2vec	0.713398 ± 0.0159	11.967
M-NMF	0.741063 ± 0.0185	1.749

4.4 Centrality

Concerning the constant increasing size of network data, the calculation of some structural properties, such as node centrality, has a high computational cost. Some algorithms provide approximation solutions using sampling and calculate the single-source shortest path for a given sample of nodes [4,24]. Even though the accuracy of these algorithms is acceptable, the computational cost is still difficult to manage [12]. Motivated by this discussion, as we find the AVPRA

less computationally complex in comparison to other state-of-the-art algorithms, we evaluated its capabilities in capturing the centrality of nodes.

For this purpose, using a standard SN analysis library[2], we computed the betweenness centrality of each node. Centrality measures are expressed as continuous values, some times normalized in the range $[0, 1]$, while in order to train a classifier we need discreet ground-truth labels. Witnessing the Populite heterogeneous structural characteristic, such as power-law degree centrality similar to other SNs, we propose three approaches for categorizing the nodes according to their betweenness values into different intervals:

- **approach (a):** Homogeneous intervals: the centrality values are divided by intervals of 0.05 and nodes are categorized accordingly.
- **approach (b):** Heterogeneous intervals: 100 intervals between 0.0005 and 0.05, 5 intervals between 0.05 and $0.1, (0.1, 0.2], (0.2, 0.5], (0.5, 1]$.
- **approach (c):** Heterogeneous Decreasing intervals: $[0, x], (x, 1.5x]$, $(1.5x, 1.5 \times 1.5 \times x], \ldots$, with $x = 0.0001$.

Following the three above-mentioned approaches, we adopt the obtained categories as the ground-truth labels. An RF classifier is then trained to predict centrality categories values based on the vectors of various graph embedding algorithms. The F1 scores of each algorithm in predicting the labels of the test set are presented in Table 4 considering 80% of data set as train set and 20% test set.

Table 4. The accuracy of learning Centrality of the Populite network leveraging the VLs resulted by AVPRA for each node by a Random Forest Classifier. Using the three categorization methods defined in Sect. 4.4, the values are reported by $mean \pm std$ for AVPRA, DeepWalk, Node2Vec and M-NMF algorithms.

	Accuracy ($mean \pm std$)		
Algorithm	approach (a)	approach (b)	approach (c)
AVPRA	**0.601307**	**0.594771**	**0.555**
DeepWalk	0.4200 ± 0.051	0.5093 ± 0.033	0.3403 ± 0.063
Node2vec	0.4365 ± 0.039	0.5100 ± 0.038	0.3439 ± 0.057
M-NMF	0.3483 ± 0.052	0.4803 ± 0.030	0.2797 ± 0.042

4.5 Label-Drift Detection

In real world scenarios, both the structural properties of networks and the distribution of features in networks are constantly changing. The dynamic changes in the patterns of connections among individuals can have major impacts on the evolution of the network states and the communication flow. The VLP approach can be effective in capturing these dynamic changes as it can consider the

[2] https://networkx.org/.

variation of states due to diffusion and accumulation of features flowing through links. For this reason, we explored the accuracy of a prediction model based on the AVPRA VLs.

In the studied scenario a dynamic evolution of the system is due by the changes in the membership of the political groups at the Parliament that the Italian Constitution considers a fundamental right of parliamentarians. The method adopted to predict these changes is based on the idea of creating VLs for political groups which is the mean of the VLs of the members of the group. If we call this new vectors VL_p, the new group of a parliamentarian x, is the vector in VL_p most similar to VL_x, where similarity is measured by cosine distance.

To test the outcome, we consider 24 different extractions of the Populite data referring to each month in data set. For each time t, predict the new groups of all the candidates whose group changed between extraction t and $t + 1$. The calculated accuracy for each time period is presented in the Table 5, the average prediction accuracy obtained is 0.64.

Table 5. Number of changes happened in the mentioned period of time and the accuracy of prediction by our algorithm

Prediction of changing the labels		
Period	Number of changes in the political groups	Prediction accuracy
2020-01/2020-02	1	1
2020-02/2020-03	2	0.5
2020-03/2020-04	1	1
2020-04/2020-05	2	1
2020-05/2020-06	3	1
2020-06/2020-07	1	1
2020-07/2020-08	1	0
2020-08/2020-09	1	1
2020-09/2020-10	1	1
2020-10/2020-11	1	1
2020-11/2020-12	6	1
2020-12/2021-01	17	0.705882
2021-01/2021-02	4	1
2021-02/2021-03	23	0.956522
2021-03/2021-04	4	0.25
2021-04/2021-05	1	0
2021-05/2021-06	17	0.058824
2021-06/2021-07	1	0
2021-07/2021-08	2	0
2021-08/2021-09	0	–
2021-09/2021-10	2	0.5
2021-10/2021-11	0	–
2021-11/2021-12	1	0
Total period	**Mean number of drifts**	**Mean accuracy**
2020-01/2021-12	2.19047	0.641304

5 Discussion

In this section, we discuss the results we achieved and underline the capabilities of AVPRA in capturing both structural and domain-specific properties of the network.

5.1 Structural Properties of the Network

The set of experiments we conducted on a variety of network analysis tasks demonstrates the ability of AVPRA in capturing the structural properties of the network. In community detection, Sect. 4.1, our algorithm ranked first, demonstrating its ability to capture the network modularity calculated by the Leiden algorithm. In clustering, Sect. 4.2, AVPRA performs in line with other graph embedding algorithms, with variations depending on the clustering algorithm used. In link prediction, Sect. 4.3, our embedder outperforms others using a reduced execution time. AVPRA outperforms the state of the art also in measuring node centrality, Sect. 4.4, a task recognized as difficult in the literature for ML procedures while initially some neural network learning are required [13]. Finally, AVPRA is partially capable to detect label-drift detection, Sect. 4.5, a problem that is clearly influenced by a variety of factors that are endogenous to the communication network of the parliamentarians.

These results are achieved using an algorithm that is full transparent and incremental. The features composing the AVPRA vectors are directly obtained from the domain features and the addition of new features will not invalidate the previous steps of the procedure. Considering the graph embedding algorithms as the encoder of network features into a latent space, we would face difficulties in retrieving some information such as the centrality of nodes using the embedded vectors and information remains abstract and hard to interpret. Graph embedding algorithms usually create vectors of a dimension of several hundreds of latent features per node in the graph. While eigenvalue-based decomposition methods give some formal guarantees on the retained network properties, random-walk-based methods are stochastic in nature and depend heavily on hyper-parameter settings.

5.2 Domain-Specific Properties of the Network

One of the key capabilities of AVPRA is embedding the network structure directly using domain-specific features. This offers great support during the interpretation of the results. To illustrate the implications we developed an analysis of the sub-communities in the network, showing the differences between the analysis developed by AVPRA and other methods.

As we discussed in Sect. 4.1, community detection in network analysis is highly relevant in realizing the properties of members based on the community they are involved in. According to the homophily principle in social science, nodes located in one community may have more common features, in other words, nodes with similar features tend to create their communities. What is observed at

the level of a network can be observed also at the level of its communities, where the members of a given community tend to form groups with a higher number of common interests or stronger affinity. Even though this important problem has been explored in the literature using different approaches and terminologies, we refer to it as sub-community detection. Getting this level of granularity reveals the complexity in the structure of the network and can offer important insight into the affinities shared by nodes in different communities.

Regarding a research question raised by Populite working group about the identification of intra-party factions, we evaluate the AVPRA output vectors in retaining the sub-communities of the network properties. In order to validate the outcome, we compare it with the state-of-the-art, InfoMap algorithm [32]. InfoMap algorithm optimizes *The Map equation*, which exploits the information-theoretic duality between finding community structure in networks and minimizing the description length of a random walker's movements on a network. InfoMap supports the two and multi-level partitioning while the core idea is similar to the Louvain algorithm. Implementing this algorithm on the Populite data set, we get the information on the network structure represented in Table 6.

A schematic view of the web-based network navigator[3] is depicted in Fig. 2. The communities are labeled according to the political groups existing in the Italian parliament. The representative nodes of each community are highlighted

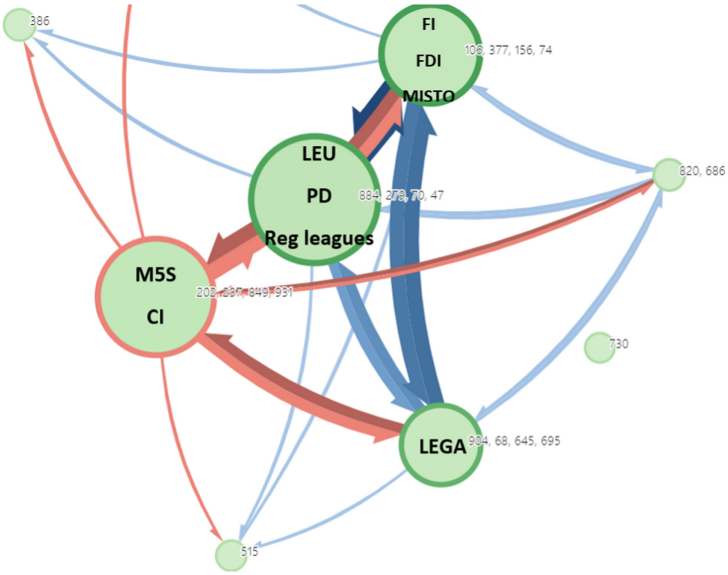


Fig. 2. Schematic view of infomap Algorithm [32] representing the Populite data set. The nodes with higher flow of information are demonstrated by each community. The information flow among each community is weighted and demonstrated by pointers.

[3] https://www.mapequation.org/infomap/.

in the figure. The thickness of links connecting the communities demonstrates the flow of information or in other words, the weight of the connection between two partitions.

Having at hand this information from InfoMap, we could find the most influential nodes in the flow of information which can be the interpretation of leaders, and nodes connected to these leaders construct the sub-communities, see Table 6 for details. On the other hand, leveraging the output vectors of AVPRA, as we discussed in the Sect. 4.2, the partitions extracted by state-of-the-art algorithms are in good accordance, more than 88%, with the partitions resulting from clustering the VLs by AVPRA, details in Table 2.

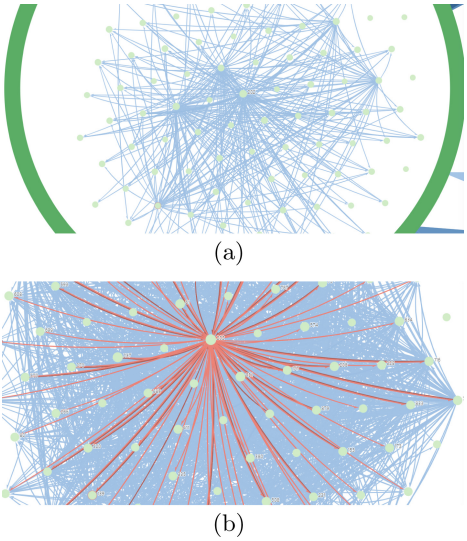
Table 6. The extracted information using the InfoMap algorithm on the Populite data set. **Flow** is the rate of received information to each communities, **in-flow** is the entering and **out-flow** is the rate of exit information. Number of nodes, links and the most involved nodes in the flow path are also mentioned.

Communities	Flow	In-flow	Out-flow	Nodes	Links	Representative nodes id
LEU/PD/REG	0.3898	0.07294	0.05730	184	11918	884,279,70,47
LEGA	0.1028	0.02777	0.03422	146	4999	904,68,645,695
FI/FDI/MISTO	0.1965	0.06379	0.06688	186	7366	106,377,156,74
M5S/CI	0.3105	0.03600	0.04211	256	15734	202,237,849,931

Considering each cluster as a community, we implement the *OPTIC* clustering algorithm on the VLs inside the partitions to find out the similarities in the second level. Table 7, presents the mean VL, VL_P, for the involved nodes inside communities and sub-communities. Clearly, the most-weighted element of VL_P for the communities, is in accordance with the one in sub-communities. This approach also could help in measuring the similarity/distances of sub-communities in terms of measuring their tendency to specific political areas. Moreover, the distribution of the existing groups and the absolute prominence of a few of them are observable. For example, in cluster 0 all sub-clusters are catheterized by a high value in the PD label. The sub-cluster 0.0 has higher values for the FDI, $M5S$, and FI labels. The sub-cluster 1 has lower values for the $M5S$ label and higher values for the IV label. Similar observations can be provided for all clusters and sub-clusters.

Table 7. The mean weight of VLs located inside a cluster(communities) and sub-clusters (sub_communities). Two examples of clusters and some of the belonging sub-clusters mean weights of elements are represented. Each element refers to a political party a node may involved in. M5S: Movimento5 Stelle; GM: Gruppo Misto; Lega; IV: Italia Viva; PD: Partito Democratico; FI: Forza Italia; FDI: Fratelli D'Italia; CI: Coraggio Italia; LU: Liberi e Uguali; Pla: Per le autonomie

	Mean weight of VL (VL_p) *of clusters/sub-clusters*									
Cluster	M5S	GM	Lega	IV	PD	FI	FDI	CI	LU	Pla
0	0.10784001	0.066733	0.055306	0.02146	**0.660737**	0.107175	0.029693	0.022745	0.036151	0.005188
Sub-clusters	***VL_p of sub-clusters***									
0	0.1162275	0.068403	0.060062	0.020338	0.646582	0.112942	0.031678	0.023786	0.036209	0.005452
1	0.09068673	0.065458	0.05509	0.022798	**0.653536**	0.110606	0.029541	0.02258	0.035431	0.00496
2	0.08138816	0.061689	0.05185	0.023376	**0.674685**	0.104717	0.027103	0.021945	0.034635	0.004913
3	0.12472662	0.068516	0.061732	0.020404	**0.637912**	0.118537	0.034373	0.024205	0.03432	0.004864
4	0.10234745	0.064337	0.045815	0.022505	**0.691376**	0.09406	0.024516	0.021495	0.035897	0.005123
Cluster										
1	**0.746649**	0.111349	0.038724	0.003204	0.075565	0.044796	0.014362	0.030933	0.009983	0.001288
Sub-clusters	***VL_p of sub-clusters***									
0	**0.748093**	0.112028	0.038639	0.00308	0.072941	0.043236	0.013958	0.031057	0.00991	0.001257
1	**0.774574**	0.112761	0.03088	0.002101	0.047472	0.031561	0.010888	0.028511	0.007825	0.000872
2	**0.774678**	0.111601	0.024984	0.003139	0.071959	0.036025	0.010138	0.029971	0.009463	0.001004
3	**0.758054**	0.11416	0.035063	0.003208	0.070387	0.038875	0.013638	0.027245	0.009758	0.001030
4	**0.78523**	0.11601	0.02202	0.01803	0.04000	0.02840	0.00817	0.03110	0.00718	0.00075



(a)

(b)

Fig. 3. Hierarchical representation of communities extracted from infomap Algorithm [32]. (a). The community of *M5S and CI* is zoomed in (b)The in-flow/out-flow of information of node 212 is depicted.

6 Conclusion

Due to the unbounded increasing size of networks representing a system of instances and interconnections, neither capturing structural nor domain-specific properties are feasible by using traditional network analysis tools. Leveraging tools to reflect domain-specific and structural properties of networks in the feature space digestible by ML algorithm for further exploration seems inevitable in real-life network studies. In our experiment, the AVPRA algorithm [37] has been proven to be able to capture the *structural properties* of the studied network on multiple *analytical tasks*, without disregarding the ability to retain its *domain-specific* features. The algorithm is also *incremental*, making it possible to update its results when the domain is evolving and supports a *transparent* analysis of the obtained results, explaining them in terms of the domain features embedded during the VLP procedure.

References

1. Abbas, A.M.: Social network analysis using deep learning: applications and schemes. Soc. Netw. Anal. Min. **11**(1), 1–21 (2021)
2. Azaouzi, M., Romdhane, L.B.: An evidential influence-based label propagation algorithm for distributed community detection in social networks. Proc. Comput. Sci. **112**, 407–416 (2017)
3. Azzini, A., et al.: Advances in data management in the big data era. In: Goedicke, M., Neuhold, E., Rannenberg, K. (eds.) Advancing Research in Information and Communication Technology. IAICT, vol. 600, pp. 99–126. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81701-5_4
4. Bader, D.A., Kintali, S., Madduri, K., Mihail, M.: Approximating betweenness centrality. In: Bonato, A., Chung, F.R.K. (eds.) WAW 2007. LNCS, vol. 4863, pp. 124–137. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77004-6_10
5. Bellandi, V., Ceravolo, P., Maghool, S., Siccardi, S.: Toward a general framework for multimodal big data analysis. Big Data (2022)
6. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech: Theory Exp. **2008**(10), P10008 (2008)
7. Bonner, S., Brennan, J., Kureshi, I., Theodoropoulos, G., McGough, A.S., Obara, B.: Evaluating the quality of graph embeddings via topological feature reconstruction. In: 2017 IEEE International Conference on Big Data (Big Data), pp. 2691–2700 (2017). https://doi.org/10.1109/BigData.2017.8258232
8. Borgatti, S.P., Halgin, D.S.: On network theory. Organ. Sci. **22**(5), 1168–1181 (2011)
9. Cao, S., Lu, W., Xu, Q.: Grarep: learning graph representations with global structural information. In: International Conference on Information and Knowledge Management (CIKM), pp. 891–900 (2015)
10. Chiesi, A.: Network analysis. In: Smelser, N.J., Baltes, P.B. (eds.) International Encyclopedia of the Social & Behavioral Sciences, Pergamon, Oxford, pp. 10499–10502 (2001). https://doi.org/10.1016/B0-08-043076-7/04211-X,https://www.sciencedirect.com/science/article/pii/B008043076704211X

11. Emirbayer, M., Goodwin, J.: Network analysis, culture, and the problem of agency. Am. J. Sociol. **99**(6), 1411–1454 (1994)
12. Grando, F., Granville, L.Z., Lamb, L.C.: Machine learning in network centrality measures: tutorial and outlook. ACM Comput. Surv. (CSUR) **51**(5), 1–32 (2018)
13. Grando, F., Lamb, L.C.: On approximating networks centrality measures via neural learning algorithms. In: 2016 International Joint Conference on Neural Networks (IJCNN), pp. 551–557. IEEE (2016)
14. Gregory, S.: Finding overlapping communities in networks by label propagation. New J. Phys. **12**(10), 103018 (2010)
15. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 855–864 (2016)
16. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Conference on Advances in Neural Information Processing Systems (NIPS), pp. 1024–1034 (2017)
17. Hasan, M.A., Zaki, M.J.: A survey of link prediction in social networks. In: Social network data analytics, pp. 243–275. Springer (2011). https://doi.org/10.1007/978-1-4419-8462-3_9
18. Jokar, E., Mosleh, M.: Community detection in social networks based on improved label propagation algorithm and balanced link density. Phys. Lett. A **383**(8), 718–727 (2019)
19. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv:1609.02907 (2016)
20. Li, Q., Zhou, T., Lü, L., Chen, D.: Identifying influential spreaders by weighted leaderrank. Phys. A **404**, 47–55 (2014)
21. Lim, M., Abdullah, A., Jhanjhi, N., Khan, M.K.: Situation-aware deep reinforcement learning link prediction model for evolving criminal networks. IEEE Access **8**, 16550–16559 (2019)
22. Martínez, V., Berzal, F., Cubero, J.C.: A survey of link prediction in complex networks. ACM Comput. Surv. (CSUR) **49**(4), 1–33 (2016)
23. McDaid, A.F., Greene, D., Hurley, N.: Normalized mutual information to evaluate overlapping community finding algorithms. arXiv preprint arXiv:1110.2515 (2011)
24. Mendonça, M.R., Barreto, A.M., Ziviani, A.: Approximating network centrality measures using node embedding and machine learning. IEEE Trans. Netw. Sci. Eng. **8**(1), 220–230 (2020)
25. Nurek, M., Michalski, R.: Combining machine learning and social network analysis to reveal the organizational structures. Appl. Sci. **10**(5), 1699 (2020)
26. Opitz, J., Burst, S.: Macro f1 and macro f1. arXiv preprint arXiv:1911.03347 (2019)
27. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 1105–1114 (2016)
28. Palmonari, M., Minervini, P.: Knowledge graph embeddings and explainable ai. Knowl. Graphs Explain. Artifi. Intell. Found. Appli. Challenges **47**, 49 (2020)
29. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 701–710 (2014)
30. Perozzi, B., Kulkarni, V., Chen, H., Skiena, S.: Don't walk, skip! online learning of multi-scale network embeddings. In: International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 258–265 (2017)

31. Rossi, A., Barbosa, D., Firmani, D., Matinata, A., Merialdo, P.: Knowledge graph embedding for link prediction: a comparative analysis. ACM Trans. Knowl. Discovery Data (TKDD) **15**(2), 1–49 (2021)
32. Rosvall, M., Bergstrom, C.T.: Maps of information flow reveal community structure in complex networks. arXiv preprint physics.soc-ph/0707.0609 (2007)
33. Salehi Rizi, F., Granitzer, M.: Properties of vector embeddings in social networks. Algorithms **10**(4), 109 (2017)
34. Silva, T.C., Zhao, L.: Machine Learning in Complex Networks. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-17290-3
35. Sun, H., Huang, J., Zhong, X., Liu, K., Zou, J., Song, Q.: Label propagation with-degree neighborhood impact for network community detection. Comput. Intell. Neurosci. **2014**, 130689 (2014)
36. Traag, V.A., Waltman, L., Van Eck, N.J.: From louvain to leiden: guaranteeing well-connected communities. Sci. Rep. **9**(1), 1–12 (2019)
37. Bellandi, V., Ceravolo, P., Damiani, E., Maghool, S.: Agent-based vector- label propagation for explaining social network structures. CCIS, vol. 1593 (2022). https://doi.org/10.1007/978-3-031-07920-7_24
38. Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., Yang, S.: Community preserving network embedding. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)
39. Xie, J., Szymanski, B.K., Liu, X.: Slpa: uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In: 2011 IEEE 11th International Conference on Data Mining Workshops, pp. 344–349. IEEE (2011)
40. Xing, Y., Meng, F., Zhou, Y., Zhu, M., Shi, M., Sun, G.: A node influence based label propagation algorithm for community detection in networks. Sci. World J. **2014**, 627581 (2014)
41. Xu, X., Yuruk, N., Feng, Z., Schweiger, T.A.: Scan: a structural clustering algorithm for networks. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 824–833 (2007)
42. Zhu, X., Ghahramani, Z.: Learning from labeled and unlabeled data with label propagation (2002)