IJE TRANSACTIONS C: Aspects Vol. 33, No. 3, (March 2020) 366-376

International Journal of Engineering

Journal Homepage: www.ije.ir

Detecting Overlapping Communities in Social Networks using Deep Learning

S. M. M. Salehi*, A. A. Pouyan

Department of Computer Engineering, Shahrood University of Technology, Shahrood, Iran

P A P E R I N F O

Paper history:
Received 14 December 2019
Received in revised form 14 January 2020
Accepted 17 January 2020

Keywords:
Community Detection
Overlapping Communities
Deep Learning
Social Networks
Graph Embedding

A B S T R A C T

In network analysis, the community is considered as a group of nodes that is densely connected with respect to the rest of the network. Detecting the community structure is important in any network analysis task, especially for revealing patterns between specified nodes. There are various approaches in literature for community, overlapping or disjoint, detection in networks. In recent years, many researchers have concentrated on feature learning and network embedding methods for nodes clustering. These methods map the network into a lower-dimensional representation space. In this paper, we propose a model for learning graph representation using deep neural networks. In this method, a nonlinear embedding of the original graph is fed to stacked auto-encoders for learning the model. Then an overlapping clustering algorithm is employed to extract overlapping communities. The effectiveness of the proposed model is investigated by conducting experiments on standard benchmarks and real-world datasets of varying sizes. Empirical results exhibit that the presented method outperforms some popular community detection methods.

***doi**: 10.5829/ije.2020.33.03c.01*

1. INTRODUCTION

In most complex systems, data can be represented as networks like transportation networks, citation networks, World Wide Web, co-authorship networks, and social networks. Analyzing these networks yields insight into their structures. Growing attention has been paid to this analysis in recent years, particularly due to the omnipresence of such networks in real-world [1].

Social network analysis (SNA) is a novel research topic in the field of computer science. The goal of SNA is to explore associations between social entities, and to figure out the features and properties of the entire network, usually by graph theory. A graph is a means of describing relationships among a set of specified items. Given a graph $G = (V, E)$, V is a set of n nodes (vertices), and E is a set of m edges (links). In a social network, nodes represent individuals, and their social interactions are presented by edges. The network structure, for unweighted networks, can be determined by an $n \times n$ adjacency matrix A. Each element A_{ij} of A is equivalent to 1 if there is an edge that connects nodes i and j. Otherwise, it is 0.

Community structure is a shared property in many social networks. Despite the lack of a unique definition for the community, a community is assumed to represent a group of nodes where the number of connections between these nodes is greater than connecting with other nodes in the network. A community is considered as a group of nodes similar to each other, but different from other nodes in the network. Individuals that are in a similar community have comparable properties [2] such as social ties, interests, occupation, location, etc.

Community structure often unravels precious information of interest. Community detection (CD) in networks indicates an exciting problem with several applications, particularly for knowledge extraction from social networks. Several CD algorithms have been presented for the division of a social network into communities [3]. Within the community detection framework, much attention is given to the identification of disjoint communities. In disjoint (separated or non-overlapping) CD, a node may belong to only one community. However, in real-world scenario, a node may belong to more than one community that leads to overlapping communities. For instance, a person may

*Corresponding Author Email: *mahdi_salehi@shahroodut.ac.ir* (S. M. M. Salehi)

Please cite this article as: S. M. M. Salehi, A. A. Pouyan, Detecting Overlapping Communities in Social Networks using Deep Learning, International Journal of Engineering (IJE), IJE TRANSACTIONS C: Aspects Vol. 33, No. 3, (March 2020) 366-376

forge connections to many social groups such as friends, family, and co-workers [4].

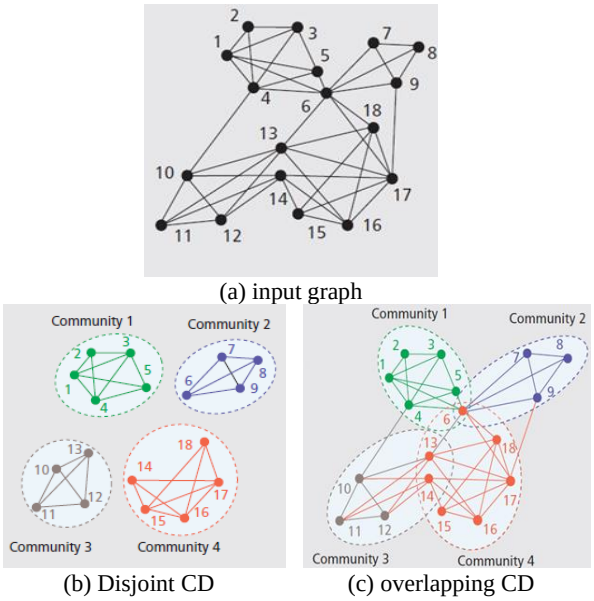
This overlap is a remarkable feature of most real-world networks. Naturally, nodes may be a member of multiple communities. Therefore, by forcing each node into a single community, we miss much information and may obtain a misleading characterization of the underlying community structure.

A good comparison is shown in Figure 1 for this problem. Figure 1a displays an undirected social graph comprising of 18 individuals and Figure 1b shows the four isolated communities of the social network from Figure 1a. Here nodes 6, 13, and 14 in Figure 1c are overlapping nodes that can be shared by two or three communities.

Over the last decade, numerous methods have been presented for the detection of community. In the following section, these methods are explained in detail. Some of these methods have a number of drawbacks or limitations including inability to support the concept of overlapping, or high costs of computation for large graphs.

In this research, we introduce an effective algorithm to address the detection of overlapping communities by incorporating the community structure into graph embedding. Our formulation combines the strengths of random walk methods and deep learning architectures in a framework. The proposed algorithm maintains both the global and local structure of a network. Experimental results exhibit the efficacy of the presented method on some benchmarks and real-world networks.

This paper is organized as follows. Section 2 reviews the literature on community detection algorithms in networks, with emphasis on overlapping methods. Section 3 explains the design of the proposed algorithm in detail. It presents preliminaries such as the concept of node representation, deep learning components, and an overlapping clustering algorithm used on deep learning output. In Section 4, we outline our experiments on some computer-generated and real-world networks. Section 5 presents the results of experiments and compares our algorithm with a few overlapping community detection algorithms. Finally, conclusion is drawn in Section 6.



(a) input graph

(b) Disjoint CD (c) overlapping CD

Figure 1. An example of a small social network; disjoint (non-overlapping) vs. overlapping community structure [5]

2. RELATED WORKS

The detection of overlapping and non-overlapping communities represents two distinct problems. That is, the algorithms used to detect non-overlapping communities could not be readily applied to other overlapping counterparts. Hence, a variety of algorithms have been introduced to exhibit the structures of overlapping communities [6, 7].

The algorithms for detecting communities in graphs can be divided into several categories based on diverse parameters, such as the actual operational methods [3], or the fundamental concept of the community [8]. In this section, we present an abridged overview of a few algorithms for community detection.

Community detection can be performed by detecting the denser areas of the network. A dense community is a group of vertices where the number of edges is substantially greater than the number of edges expected in a random graph (without any community structure). With this approach, community detection can be considered as optimizing a specific density function until no further augmentation is possible. Modularity [9] is a famous density function that is used to detect disjoint communities. Chen *et al.* [10] presented an extended version of modularity for overlapping community detection.

Communities could be assumed as parts of the graph obtained by removing all edges (i.e., bridges) that connect these parts to each other. Girvan-Newman (GN) algorithm is a famous method in this category [11]. It conducts hierarchical clustering (dividing a network into preferred numbers of clusters). GN relies on a centrality measure called betweenness. At each step, it involves counting the shortest paths between all pairs of vertices that exist in the network, sorting edges by their betweenness values, and removing edges with the most value (i.e., a bridge). Cluster Overlap Newman Girvan Algorithm (CONGA) extends GN for discovering overlapping communities in networks [12]. A central point of CONGA is the concept of split betweenness, which allows a node to split into multiple copies.

A community could be defined as a set of nodes categorized by the dissemination of a similar property, action, or data within the network. The label propagation algorithm (LPA) deals with this definition, in which

nodes that belong to an identical label forge a community [13]. A node in the network decides to join a community where the majority of its neighbors are members of that community. With the propagation of labels, groups of nodes with tight connections obtain an exclusive label. This method is fast but does not support the overlapping concept. In the community overlap propagation algorithm (COPRA), nodes have a label and a belonging coefficient [14]. The belonging coefficient of each node is updated by averaging coefficients in all of its neighbors. A node that has an equivalent maximum number of neighbors, in at least two communities, is considered as an overlapping node. The LPA can be extended to an overlapping community detection by Speaker-listener Label Propagation Algorithm (SLPA) [15]. It gives each node a memory for the storage of the received information and the spread of labels between nodes in compliance with rules of pairwise interaction. The membership strength is defined as the possibility of complying with a label in node memory. The output of this method is the labels' probability distribution for each node's memory. Nodes with identical labels are then assigned to a community. A node with several labels belongs to several communities.

Another common approach to community detection defines it as an accurate structure of edges. Therefore, an algorithm defines some sort of structure and then attempts to find it efficiently within the graph. For instance, a clique is defined as a complete subgraph of a given graph. The clique of the largest possible size is called a maximal clique. The clique percolation method (CPM) [16] finds all k-cliques (clique with k nodes) in a graph and merges cliques with common k-1 nodes. A community embodies the union of all k-cliques that can be reached from each other. GCE (greedy clique expansion) [17] identifies maximum cliques as seed communities, which are then expanded through greedy optimization of a local fitness function.

Another idea that has been explored is to partition links (edges) rather than nodes for exploring community structure. According to this approach, it is the relation, not the node, which is a member of the community. Hence, since they try to cluster the network edges, the nodes belong to a set of communities forged by their edges. Link community (LC) is a well-known method in this category [18]. In LC, an original graph's node is said to be overlapping when its connecting links are placed in at least two or more clusters.

In recent years, the methods that rely on the representation of networks in vector space (mapping graph matrices to another space) have become popular. Graph embedding seeks to offer a representation of a graph as low dimensional vectors but preserves the graph structure [19]. Finding a vector representation for nodes of a graph is challenging and poses several difficulties. A suitable vector representation of nodes needs to retain the network structure and the connection between single nodes. The network structure is represented by proximity measures such as first-order proximity (edge weights), second-order proximity (similarity of the vertices' neighborhood structures), and higher-order proximities between nodes.

The embedding methods can be split into three general categories: methods based on (1) Factorization (2) Random Walk, and (3) Deep Learning (DL).

In the first category, connections of nodes are represented as a matrix, which is then factorized to attain the embedding. The matrices employed for the representation of the connections comprise the Laplacian matrix, Katz similarity matrix, etc. Graph Factorization GraRep [20] is an example of this category.

Several graph properties such as node similarity and centrality are approximated using random walks. They are particularly useful when only partial observation of a graph is possible. Deepwalk [21] is a method that transforms graphs into collections of linear sequences by a uniform sampling method known as the truncated random walk.

Deep learning has been particularly successful in handling images, texts, time series, and audio data in recent years [22]. Thanks to the advances in deep neural networks, these methods can produce representations for various kinds of data [23]. Deep learning can be used as a useful tool for graph analysis to produce representations that contribute to community detection.

3. PROPOSED METHOD

We propose an algorithm to investigate the problem of overlapping community detection. The overall scheme of the proposed method is shown in Figure 2.

The algorithm exploits factorized matrices, generated random walks, and deep learning architecture. It consists of three major parts: First, we introduce a model based on random walks to represent (embed) the nodes. This model captures the input graph's structural information. It generates probabilistic co-occurrence (PCO) matrix and positive pointwise mutual information (PPMI) matrix. After that, a deep learning architecture is used to learn representations of low-dimensional nodes. The output of this part is given to an overlapping clustering method to prodive overlapping communities. In this section, the major components of the proposed algorithm are discussed in detail.

3. 1. Graph Embedding As mentioned earlier, graph (node) embedding aims to offer a representation of a graph (node) as low dimensional vectors while retaining the structure of the network. As the output of this process, any node is described with a vector.

Pioneer methods for graph embedding generalize advancements in language-modeling (sequences of words) to graphs (sequences of nodes) for representation.

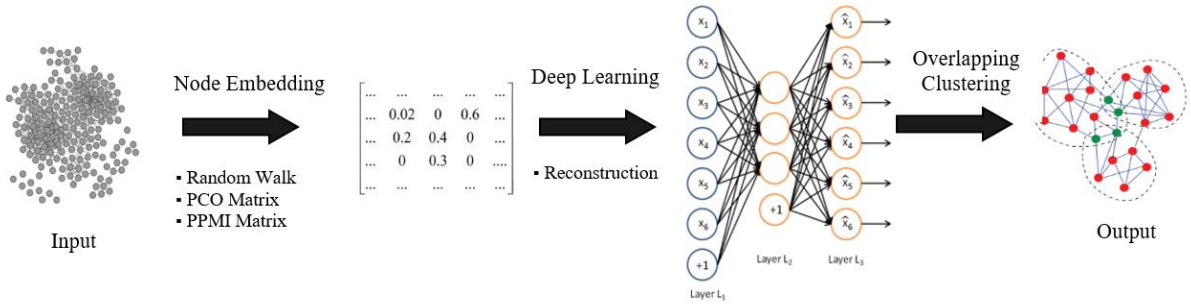


Figure 2. The framework of the method presented in the study

The essential inputs of any language-modeling algorithm are vocabulary and a corpus. The language-modeling aims to estimate the probability of a sequence of words manifesting in a corpus [24]. Formally put, for a sequence of words in Equation (1) which w_i is an element of vocabulary:

$$W_1^n = (w_0, w_1, \ldots, w_n) \quad (1)$$

We would like to maximize the probability in Equation (2) over the entire corpus.

$$\Pr(w_n \mid w_0, w_1, \ldots, w_{n-1}) \quad (2)$$

It is similar to calculating the probability of observing vertex v_i based on all previously visited vertices in a random walk:

$$\Pr(v_i \mid v_1, v_2, \ldots, v_{i-1}) \quad (3)$$

Based on this analogy, Deepwalk [21] presented a generalization of language-modeling to examine the graph via a set of random walks. These walks could be considered as short phrases and sentences in a certain language. Deepwalk draws on local information attained from curtailed random walks. The procedure involves a slow uniform sampling process, and hyper-parameters of the method such as "walk length" and "walks per node" are not easy to determine.

Our algorithm directly constructs The PCO matrix from a graph, which avoids the expensive sampling process. A co-occurrence matrix is a giant matrix as wide as the vocabulary (graph) size. When words (nodes) occur together, they are marked with a positive entry. Otherwise, they will take a 0.

Inspired by Google's PageRank, we consider a random walk model with a restart. First, the vertices are randomly ordered in a graph. It is assumed that the current vertex is the i^{th} vertex, and there exists a transition matrix T that takes into account the transition likelihoods between diverse vertices. Matrix T is initialized based on the adjacency matrix.

Each row vector of the PCO matrix is of the form p_k where the j^{th} entry corresponds to the likelihood of reaching the j^{th} vertex following k steps of transition. According to probability α, a random walk will continue (choosing the next node), and based on probability $1 - \alpha$, it will revert to the original vertex and the procedure will be restarted. The probabilities for arriving at different vertices after exactly k steps of transitions is specified by recurrent Equation (4):

$$p_k = \alpha . p_{k-1} T + (1-\alpha) p_0 \quad (4)$$

P_0 is the initial one-hot vector of a node v_i with the i^{th} entry having a value of 1, and other entries having a value of 0. If there is no random restart in the procedure (α=1), PCO is achieved using Equation (5):

$$\begin{aligned} p_k^* &\triangleq p_k \big|_{\alpha=1} \\ &= p_{k-1}^* T = p_0 T^k \end{aligned} \quad (5)$$

After constructing the PCO matrix, the representation r for the i^{th} node is shown in Equation (6):

$$r = \sum_{k=1}^{K} f(k) . p_k^* \quad (6)$$

The $f(\cdot)$ is a decreasing weighting function. The weight assigned to a context node is a function of the distance of that node from the current node in the graph. In our method, we use $f(x) = 1/x$ as a common weighting function [25].

In language-modeling, we use the current word w (whose representation that we want to generate) to predict its c context in a probabilistic form. The term 'context' refers to the words that may appear around the current word in a sentence. With K as window size, sentences are generated using K words before and after the current word. Following this, the PCO matrix is used to compute the PMI matrix. The PMI matrix can be seen as the product of the representation matrix and the context matrix, which could be expressed by Equation (7):

$$PMI_{w,c} = \log\left(\frac{\#(w,c).|L|}{\#(w).\#(c)}\right) \quad (7)$$

where |L| is computed using Equation (8):

$$|L| = \sum_{w}\sum_{c} \#(w, c) \tag{8}$$

In the same vein, for graph embedding, v_i in Equation (9) refers to the current node and W_{v_i} determines the nodes that appear in a random walk rooted at v_i.

$$PMI_{v_i, W_{v_i}} = \log\left(\frac{\#(v_i, W_{v_i}).|L|}{\#(v_i).\#(W_{v_i})}\right) \tag{9}$$

To improve performance, we use a PPMI matrix instead of a PMI matrix [26]. As shown in Equation (10), each negative value is assigned to zero.

$$PPMI_{v_i, W_{v_i}} = \max(PMI_{v_i, W_{v_i}}, 0) \tag{10}$$

Each node's global neighborhood is expressed by a row of PPMI. For any two nodes, the PPMI values answer the question of whether the two nodes tend to occur more together or more independently.

Instead of factorized matrices used in our proposed method, Deepwalk combines a random walk and skip-gram model utilized in Word2vec [27] to learn network representations. Skip-gram is an approach that seeks to obtain similar vectors for similar words (based on their context). We compare the results in Section 5.

In some methods, Singular Value Decomposition (SVD) is used for dimensionality reduction after computing the PPMI matrix for a text corpus [26] or graph embedding [20]. SVD decomposes any specified matrix into three matrices (two orthonormal matrices and one diagonal matrix). Although SVD is an effective tool for dimensionality reduction, it also captures linear relationships between the two vectors. In our model, we replaced SVD with deep neural networks to capture a non-linear mapping between the nodes.

Inputting the PPMI matrix guarantees that the deep learning architecture can account for the proximity of the input graph's nodes.

3. 2. DL Components and Configuration Based on unsupervised learning, deep learning employs multiple layers of processing to learn data representations with multiple abstraction levels.

An autoencoder (AE) is a form of neural network that learns features to reconstruct its input at the output layer. In [28], autoencoders are used for graph clustering tasks with the Laplacian matrix as their input. Here, the PPMI matrix was used as an input fed to the autoencoder.

An autoencoder comprises two parts: the encoder, and the decoder. The former is responsible for embedding the original input into a low-dimensional representation. The decoding layers will learn how to decode the representation and brings it back into its original form in the most accurate manner [29]. Stacked autoencoders (SAE) consist of multiple layers of such autoencoders for learning multiple layers of representations (Figure 3).

The embedding for a node in the PPMI matrix will be a $1 \times n$ vector. This is fed to the autoencoder to reduce the dimensions to $1 \times d$ $(d << n)$ during the encoding step. We then apply function $f(\cdot)$ to vector x in the input space so that it can be sent to a new feature space.

$$f_{\theta_1}(x) = \sigma(W_1 x + b_1) \tag{11}$$

This result is called a code or latent variable. The term $\theta_1 = (W_1, b_1)$ defines the parameters involved in the encoder part, where W_1 is a weight matrix and b_1 is a bias vector. An activation function $\sigma(.)$ is usually recruited to offer for modeling of the non-linearities between two vector spaces. For this purpose, we use a half-wave rectifier $f(z) = max(z,0)$ known as rectified linear unit (ReLU). It typically learns very fast in networks with many layers.

The decoding step involves using a reconstruction function $g(\cdot)$ to reconstruct the original input vectors retrieved from the latent representation space.

$$g_{\theta_2}(y) = \sigma(W_2 y + b_2) \tag{12}$$

In Equation (12), y is the output of the encoding step, and θ_2 defines the parameters involved in the decoder (W_2 as weight matrix and b_2 as a bias vector).

Equation (13) expresses the reconstruction error as a measure of discrepancy between input x and its reconstruction in output.

$$\sum_{i} L\left(x^{(i)}, g_{\theta_2}(f_{\theta_1}(x^{(i)}))\right) \tag{13}$$

L is a loss function such as cross-entropy or mean square error (MSE). Thus, our goal is to reduce the reconstruction error by finding values of θ_1 and θ_2.

We want to estimate parameters and hyper-parameters (number of layers, or neurons in each layer, etc.) to have a precise reconstruction. This process requires effort and expertise. Obviously, the output layer has n neurons such as the input layer. The encoder and decoder parts are symmetric to each other regarding the number of layers and the number of neurons in each layer.

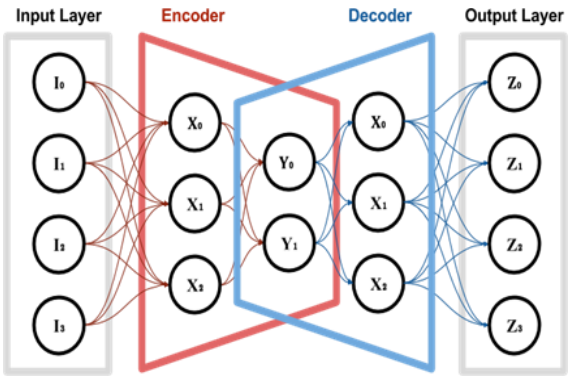


Figure 3. Structure of an SAE: an output layer, an input layer, and three hidden layers (encoding and decoding parts) [29]

Instead of randomly initializing the weights, we use the Glorot-uniform method (Xavier method), which has much better performance. Stochastic gradient descent (SGD), with a learning rate set to 1% at the beginning of the training, is used for backpropagation.

As a regularization technique for reducing overfitting, Dropout (randomly drops some neurons together with connections formed in the neural network during training phase) is used with 0.1 as drop rate. For implementation, we use the Python deep learning library Keras.

3. 3. Overlapping Clustering Algorithm K-means is a clustering algorithm that is extensively used. It leads to the detection of a disjoint community where a user is only the member of a given community. We use a version of overlapping K-means (OKM) [30] that generalizes the k-means algorithm and applies it to the output of the DL part. OKM defines an objective function for multi-assignment, initializing as data image an arbitrary cluster prototype that has a random centroid. It uses input objects along with the prototype to estimate the average of the two vectors that are used as a threshold for assignment of objects to multiple clusters (communities). The exact number of clusters must be determined as a priori.

4. EXPERIMENTS

We conduct our experiments on two datasets of varying sizes and characteristics: real-world datasets and synthetic datasets (benchmarks).

First, the details of three publicly available real-world networks and five benchmarks are given. Then, four popular metrics (F-Score, Normalized Mutual Information (NMI), conductance, and Omega index) are reviewed. The performance of our proposed method can be evaluated in comparison with some advanced algorithms to detect overlapping communities.

4. 1. Datasets Three datasets from real-world networks are selected from the KONECT dataset collection [31] with different structures. The Advogato is an online trust community platform designed for free software developers. The Hamsterster is the site of friendships for users of the website hamsterster.com. The Virgili is the email communication network utilized at the University of Virgili in southern Spain. Each edge suggests that more than one email has been sent.

Table 1 summarizes the characteristics of these datasets. The size of each dataset is equal to n (number of nodes), m indicates the number of edges, k_{avg} is the average degree, and k_{max} is the maximum degree of nodes. Obviously, k_{max} is much greater than k_{avg}. This is a common attribute of social networks, as an instance of scale-free networks [32], which follows the power-law distribution for its node degree. This distribution is expressed as $\rho x^{-\tau}$, where ρ is a constant factor, x is the variable of distribution, and τ is the exponent. If the degree distribution of a connected graph complies the power law, it leads to a long tail of intermittent vertices (many low-degree nodes vs. a few high-degree nodes).

The parameter n_{GCC} shows the greatest connected component of the graph (a maximal subset of nodes that are all reachable from each other). For a connected graph with one component, n_{GCC} is equal to n.

The density is a measure of how tightly interconnected a graph, which is calculated as the ratio of edges to the total number of possible edges. The diameter (D) is a measure of the distance between the two most distant nodes in the network. It is the maximum number of connections required to traverse the graph.

TABLE 1. Characteristics and DL settings of real-world datasets

Dataset	n	m	k_{avg}	k_{max}	n_{GCC}	Density	D	CC	Layer Configuration
Advogato	6541	51127	15.63	943	5042	0.0011	9	0.0922	6541-1024-512-256-128-64
Hamsterster	1858	12534	13.49	272	1788	0.072	14	0.0904	1858-512-256-128-64
Virgili	1133	5451	9.62	71	1133	0.0085	8	0.1660	1131-512-256-128-64

TABLE 2. Characteristics and DL settings of benchmarks

Dataset	n	m	k_{avg}	k_{max}	μ	min_C	max_C	O_m	O_n	Layer Configuration
Bench1	250	731	5	50	0.1	10	25	2	0.1	250-128-64-32
Bench2	500	1413	5	50	0.1	20	100	2	0.1	500-128-64-32
Bench3	1000	7685	10	50	0.1	20	100	2	0.1	1000-512-256-128-64
Bench4	2000	15488	10	100	0.1	25	250	3	0.2	2000-512-256-128-64
Bench5	5000	37769	15	250	0.2	50	500	4	0.3	5000-1024-512-256-128-64

By definition, the clustering coefficient (*CC*) for a node is the number of edges that exist between neighbors of that node, divided by the number of possible edges that can exist between its neighbors. The graph's clustering coefficient indicates the mean clustering coefficient of all its nodes.

The number of neurons existing in the input layer and encoding part of DL (only for the first half) is specified in the last column of Table 1. The DL configuration can be obtained by adding the decoder part. For example, configuration for Virgili is 1131-512-256-128-64-128-256-512-1131.

We use another kind of dataset for our experiments. The LFR benchmark [33] is an algorithm that produces benchmark networks (realistic artificial networks that correspond to networks in the real world). LFR presents a set of parameters for controlling the network topology. It assumes that both the degree and size of community distributions in a network follow power laws displayed by exponents' τ_1 and τ_2, respectively. We generate five datasets called Bench1 to Bench5 with $\tau_1=2$ and $\tau_2=1$.

Unlike the real world datasets, all the parameters are adjusted by the user (except *m*). In Table 2, μ is defined as the mixing parameter. It controls the fraction of edges between communities. There is a common fraction ($1-\mu$ of its edges) between each node and other nodes in its community as well as a fraction μ and other nodes in the network. Here, two parameters of max_c and min_c are the maximum and minimum community size, respectively.

Two parameters O_m and O_n adjust the overlapping properties. O_m is for overlapping diversity, as the number of community memberships in each overlapping node. O_n stands for overlapping density, as the fraction of overlapping nodes in the graph. For a disjoint community detection algorithm, $O_m = 1$ and $O_n = 0$.

4. 2. Metrics To evaluate performance of the proposed method, we compare it with other community detection algorithms against four popular metrics such as F-score, NMI, modularity, and conductance.

F-score measures the matching between communities obtained by a method, with communities of ground truth (a predefined expected categorization). It can be defined as the harmonic average of precision and recall.

$$F = \frac{2.precision.recall}{precision + recall} \tag{14}$$

In overlapping nodes, precision is defined as the proportion of accurately detected overlapping nodes to the entire number of detected nodes. The recall for overlapping nodes is defined as the proportion of accurately detected overlapping nodes to the entire number of overlapping nodes. F-score ranges from 0 to 1, with higher values, indicate a larger degree of correspondence. The ground truth is accessible in benchmark networks, so we can apply metrics such as F-score and NMI to them.

The NMI, introduced in [34], is another popular metric to evaluate the quality of a community detection algorithm. It measures the similarity between a reference (ground truth) model of communities and the community detection obtained by an algorithm.

It assessed the extent of resemblance between a cover (set of overlapping communities) as the output of a CD algorithm, with the ground truth cover. The mutual information *I(X,Y)* is defined in Equation (15).

$$\begin{aligned} I(X,Y) &\triangleq \sum_x \sum_y P(x,y) \log \frac{P(x,y)}{P(x)P(y)} \\ &= H(X) - H(X \mid Y) \end{aligned} \tag{15}$$

where x_i and y_i denote the community labels of node *i* in covers $\dot{X}$ and $\dot{Y}$, respectively. Labels *x* and *y* indicate the values of two randomly selected variables *X* and *Y*. In addition, *I(X,Y)* shows how much can be learned about *X* if *Y* is known, and vice versa. Based on *I(X,Y)*, and *H(X)* as the entropy of *X*, NMI is defined in Equation (16). The NMI value ranges from 0 to 1, with a higher value serving as a sign of better performance.

$$I_{norm}(\dot{X},\dot{Y}) = \frac{2I(X,Y)}{H(X) - H(Y)} \tag{16}$$

Modularity function is the most commonly used measure to evaluate the quality of a partition community structure (for non-overlapping communities). Modularity quantifies if a set of entities is more connected than a null model (randomly rewired nodes with no community structure) and thus can be considered as a community.

An extended modularity function [10] for overlapping community detection is shown in Equation (17):

$$Q_{ov} = \frac{1}{2m} \sum_{c \in C} \sum_{i,j \in c} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \frac{1}{O_i O_j} \tag{17}$$

In this version, the number of communities that a node belongs to is used as a weight for *Q*. A_{ij} is entering the adjacency matrix for nodes *i* and *j*, *c* is a community in cover *C*, k_i denotes the degree of node *I*, and O_i indicates the number of communities to which node *i* belongs. A higher *Q* value for a partition (or cover), indicates a superior corresponding community structure.

Conductance is used as a simple measure to capture the community goodness [35]. It corresponds most closely to the intuition that a community comprises a set of densely linked nodes that are sparsely linked to the outside. The conductance for community *S* in cover *C* is given in Equation (18):

$$cond(S) = \frac{c_S}{2m_S + c_S} \tag{18}$$

where m_S shows the number of internal edges (both nodes of the edge belong to *S*) and c_S indicates the number of boundary edges (only one node of the edge belongs to *S*). Particularly, the sets of nodes that closely resemble a community are characterized by a lower conductance. It

has many inward edges and/or few edges pointing outside. The conductance of a cover is the average conductance of all its communities.

5. RESULTS AND DISCUSSION

Testing an algorithm means the analysis of a network that has a community-like structure and the recovery of its communities. In this section, performance of the presented method is empirically tested (with two values for α), in comparison with several algorithms, on networks of LFR benchmark that are characterized with real world datasets and overlapping communities.

For algorithms that have tunable parameters, we reported the results for the best setting. Considering the non-deterministic nature of COPRA and SLPA, each of them was repeated five times on each network instantiation. The maximum for the average degree in benchmarks is maintained at k_{avg} = 15, which resembles large social networks in terms of the order.

The results of applying the methods on the datasets are shown in Tables 3 to 6 for F-score, NMI, modularity, and conductance, respectively. Numbers in bold demonstrate the optimum performance in each row. We set α to *0.95* and *0.975* in our method. With $\alpha=1$, the contextual information of the nodes is not weighted based on their distance, and we obtain undesirable results.

A problem is that for real world networks, the ground truth of overlapping communities is not accessible. Since the reference communities for ground truth of these networks need to be manually determined, they are subjective in nature. For this reason, only benchmark datasets are used for metrics such as F-score and NMI in Tables 3 and 4.

Empirical results on datasets of varying structures and sizes exhibit that the presented model excels with other advanced algorithms in the overlapping community detection tasks. The results indicate that the performance of each method worsens when the graph size increases or the communities are more similar to each other. Also, the performance worsens with the increase in O_m, O_n, or μ in the benchmarks.

In Table 3, the behavior of the algorithms is studied in terms of F-score. The worst results go with LC and GCE. The clique-like assumption of GCE yields high precision, but this method has a very low recall. On the other hand, LC has a poor performance in terms of F-score despite its extremely high recall. This is because the overlapping nodes claimed by LC is more than the exception (resulting in a very low precision).

As Table 4 shows, the values of NMI for networks that have a small-sized community s = (5, 50) are usually greater than those for networks that have a large-sized community s = (15,250).

Modularity maximization forces small communities into larger ones (resolution limit). Thus, it does not detect communities that are smaller than the resolution limit.

As shown in Table 6, conductance is capable of identifying well-separated communities (low c_S and/or high m_S), but it has poor performance in identifying cohesive and dense sets of nodes with high clustering coefficients.

The results from Tables 3 to 6 reveal the merits of our proposed model by using the weighting strategy in the random walk procedure and the effectiveness of deep architectures.

TABLE 3. Results for community detection algorithms (**F-score** as performance measure)

	CONGA	COPRA	SLPA	GCE	LC	Deepwalk	Proposed (α=0.95)	Proposed (α=0.975)
Bench1	0.3213	0.3622	0.3757	0.2841	0.3092	0.4219	0.4710	**0.5118**
Bench2	0.3185	0.3370	0.3510	0.2380	0.2738	0.3772	0.4043	**0.4297**
Bench3	0.2957	0.3149	0.3278	0.2147	0.2409	0.3408	0.3721	**0.3818**
Bench4	0.2753	0.2792	0.2792	0.1975	0.2140	0.3087	**0.3514**	0.3379
Bench5	0.2314	0.2514	0.2541	0.1712	0.1946	0.2749	**0.3109**	0.3049

TABLE 4. Results for community detection algorithms (**NMI** as performance measure)

	CONGA	COPRA	SLPA	GCE	LC	Deepwalk	Proposed (α=0.95)	Proposed (α=0.975)
Bench1	0.3818	0.4417	0.4713	0.3619	0.3118	0.4629	0.5012	**0.5369**
Bench2	0.2823	0.3742	0.3921	0.3654	0.2673	0.3376	0.4090	**0.4229**
Bench3	0.2492	0.2944	0.2783	0.2504	0.2138	0.2887	0.3239	**0.3498**
Bench4	0.1793	0.2286	0.2438	0.2141	0.1782	0.2381	0.2612	**0.2779**
Bench5	0.1627	0.1940	0.2073	0.1807	0.1437	0.2028	0.2340	**0.2395**

TABLE 5. Results for community detection algorithms (**modularity** as performance measure)

	CONGA	COPRA	SLPA	GCE	LC	Deepwalk	Proposed (α=0.95)	Proposed (α=0.975)
Advogato	0.1302	0.1275	0.1548	0.1184	0.1033	0.1670	**0.1594**	0.1562
Hamsterster	0.1608	0.1543	0.1705	0.1449	0.1501	0.1836	0.1973	**0.2028**
Virgili	0.1758	0.1743	0.1810	0.1538	0.1594	0.1877	0.2190	**0.2274**
Bench1	0.2789	0.2865	0.3105	0.2419	0.2377	0.3019	0.3217	**0.3245**
Bench2	0.2566	0.2597	0.2873	0.2187	0.2041	0.2791	0.3025	**0.3092**
Bench3	0.2271	0.2289	0.2547	0.1940	0.1897	0.2492	0.2691	**0.2751**
Bench4	0.1625	0.1586	0.1749	0.1307	0.1248	0.1704	**0.2036**	0.2004
Bench5	0.1294	0.1160	0.1672	0.1037	0.0973	0.1662	**0.1893**	0.1854

TABLE 6. Results for community detection algorithms (**conductance** as performance measure)

	CONGA	COPRA	SLPA	GCE	LC	Deepwalk	Proposed (α=0.95)	Proposed (α=0.975)
Advogato	0.0834	0.1408	0.0702	0.2148	0.2931	0.0997	0.0541	**0.0510**
Hamsterster	0.0507	0.1003	0.0615	0.1229	0.1309	0.0626	0.0692	**0.0630**
Virgili	0.0459	0.0992	0.0603	0.1107	0.1194	0.0578	0.0254	**0.0237**
Bench1	0.0094	0.0251	0.0142	0.0297	0.0309	0.0114	0.0069	**0.0051**
Bench2	0.0186	0.0493	0.0280	0.0587	0.0601	0.0338	0.0120	**0.0093**
Bench3	0.0457	0.0784	0.0457	0.0945	0.0938	0.0502	0.0314	**0.0278**
Bench4	0.0676	0.1055	0.0530	0.1395	0.1487	0.0760	**0.0480**	0.0489
Bench5	0.0910	0.1602	0.0886	0.2309	0.2411	0.1033	**0.0527**	0.0554

6. CONCLUSION

In this study, we presented a deep graph representation model that deals with overlapping community detection. Our model combined the advantages of factorized matrices, random walks, and stacked autoencoders in extracting information and generating informative representations. We investigated on real world datasets and benchmarks in different tasks and showed that the proposed model outperforms several advanced algorithms. We conclude that exploring the underlying structure of data via deep learning can lead to improved representations for graphs.

7. REFERENCES

1. Huang, F., Li, X., Zhang, S., Zhang, J., Chen, J. and Zhai, Z., "Overlapping community detection for multimedia social networks", ***IEEE Transactions on Multimedia***, Vol. 1, No. 99, (2017), 1-3.
2. Mortazavi, R. and Erfani, S., "An effective method for utility preserving social network graph anonymization based on mathematical modeling", ***International Journal of Engineering, Transactions A: Basics***, Vol. 31, No. 10, (2018), 1624-1632.
3. Fortunato, S. and Hric, D., "Community detection in networks: A user guide", ***Physics Reports***, Vol. 659, (2016), 1-44.
4. Chintalapudi, S.R. and Prasad, M.K., "Mining overlapping communities in real-world networks based on extended modularity gain", ***International Journal of Engineering-Transactions A: Basics***, Vol. 30, No. 4, (2017), 486-492.
5. Yang, S., Yang, X., Zhang, C. and Spyrou, E., "Using social network theory for modeling human mobility", ***IEEE Network***, Vol. 24, No. 5, (2010), 6-13.
6. Harenberg, S., Bello, G., Gjeltema, L., Ranshous, S., Harlalka, J., Seay, R., Padmanabhan, K. and Samatova, N., "Community detection in large-scale networks: A survey and empirical evaluation", ***Wiley Interdisciplinary Reviews: Computational Statistics***, Vol. 6, No. 6, (2014), 426-439.
7. Xie, J., Kelley, S. and Szymanski, B.K., "Overlapping community detection in networks: The state-of-the-art and comparative study", ***ACM Computing Surveys (csur)***, Vol. 45, No. 4, (2013), 43.
8. Coscia, M., Giannotti, F. and Pedreschi, D., "A classification for community discovery methods in complex networks", ***Statistical Analysis and Data Mining***, Vol. 4, No. 5, (2011), 512-546.
9. Clauset, A., Newman, M.E. and Moore, C., "Finding community structure in very large networks", ***Physical review E***, Vol. 70, No. 6, (2004), 066111.
10. Chen, M., Kuzmin, K. and Szymanski, B.K., "Extension of modularity density for overlapping community structure", in IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)., (2014), 856-863.
11. Girvan, M. and Newman, M.E., "Community structure in social and biological networks", ***Proceedings of the National Academy of Sciences***, Vol. 99, No. 12, (2002), 7821-7826.
12. Gregory, S., "An algorithm to find overlapping community structure in networks", in European Conference on Principles of

Data Mining and Knowledge Discovery, Springer., (2007), 91-102.

13. Raghavan, U.N., Albert, R. and Kumara, S., "Near linear time algorithm to detect community structures in large-scale networks", ***Physical Review E***, Vol. 76, No. 3, (2007), 036106.
14. Gregory, S., "Finding overlapping communities in networks by label propagation", ***New Journal of Physics***, Vol. 12, No. 10, (2010), 103018.
15. Xie, J. and Szymanski, B.K., "Towards linear time overlapping community detection in social networks", in Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer., (2012), 25-36.
16. Palla, G., Derényi, I., Farkas, I. and Vicsek, T., "Uncovering the overlapping community structure of complex networks in nature and society", ***Nature***, Vol. 435, No. 7043, (2005), 814-818.
17. Lee, C., Reid, F., McDaid, A. and Hurley, N., "Detecting highly overlapping community structure by greedy clique expansion", the 4th Workshop on Social Network Mining and Analysis (2010), 33-42. .
18. Evans, T. and Lambiotte, R., "Line graphs, link partitions, and overlapping communities", ***Physical Review E***, Vol. 80, No. 1, (2009), 016105.
19. Goyal, P. and Ferrara, E., "Graph embedding techniques, applications, and performance: A survey", ***Knowledge-Based Systems***, Vol. 151, (2018), 78-94.
20. Cao, S., Lu, W. and Xu, Q., "Grarep: Learning graph representations with global structural information", in Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, ACM., (2015), 891-900.
21. Perozzi, B., Al-Rfou, R. and Skiena, S., "Deepwalk: Online learning of social representations", in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM., (2014), 701-710.
22. Khatami, A., Babaie, M., Tizhoosh, H., Nazari, A., Khosravi, A. and Nahavandi, S., "A radon-based convolutional neural network for medical image retrieval", ***International Journal of Engineering-Transactions C: Aspects***, Vol. 31, No. 6, (2018), 910-915.
23. LeCun, Y., Bengio, Y. and Hinton, G., "Deep learning", ***Nature***, Vol. 521, No. 7553, (2015), 436-444.
24. Mikolov, T., Chen, K., Corrado, G. and Dean, J., "Efficient estimation of word representations in vector space", ***arXiv preprint arXiv:1301.3781***, (2013).
25. Pennington, J., Socher, R. and Manning, C., "Glove: Global vectors for word representation", in Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)., (2014), 1532-1543.
26. Levy, O. and Goldberg, Y., "Neural word embedding as implicit matrix factorization", in Advances in neural information processing systems., (2014), 2177-2185.
27. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., "Distributed representations of words and phrases and their compositionality", in Advances in Neural Information Processing Systems, (2013), 3111-3119.
28. Tian, F., Gao, B., Cui, Q., Chen, E. and Liu, T.-Y., "Learning deep representations for graph clustering", in AAAI., (2014), 1293-1299.
29. Baldi, P., "Autoencoders, unsupervised learning, and deep architectures", in Proceedings of ICML Workshop on Unsupervised and Transfer Learning., (2012), 37-49.
30. Cleuziou, G., "An extended version of the k-means method for overlapping clustering", in Pattern Recognition, 2008. ICPR 2008. 19th International Conference on, IEEE,(2008), 1-4.
31. Kunegis, J., "Konect: The koblenz network collection", in Proceedings of the 22nd International Conference on World Wide Web, ACMe, (2013), 1343-1350.
32. Barabási, A.-L., "Scale-free networks: A decade and beyond", ***Science***, Vol. 325, No. 5939, (2009), 412-413.
33. Lancichinetti, A. and Fortunato, S., "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities", ***Physical Review E***, Vol. 80, No. 1, (2009), 016118.
34. Danon, L., Diaz-Guilera, A., Duch, J. and Arenas, A., "Comparing community structure identification", ***Journal of Statistical Mechanics: Theory and Experiment***, No. 09, (2005), P09008.
35. Leskovec, J., Lang, K.J. and Mahoney, M., "Empirical comparison of algorithms for network community detection", in Proceedings of the 19th international conference on World wide web, ACM,(2010), 631-640.

Detecting Overlapping Communities in Social Networks using Deep Learning

S. M. M. Salehi, A. A. Pouyan

Department of Computer Engineering, Shahrood University of Technology, Shahrood, Iran

P A P E R I N F O

Paper history:
Received 14 December 2019
Received in revised form 14 January 2020
Accepted 17 January 2020

Keywords:
Community Detection
Overlapping Communities
Deep Learning
Social Networks
Graph Embedding

چکیده

در تحلیل شبکه‌ها، مفهوم انجمن به تعدادی از گره‌های گراف اطلاق می‌شود که چگالی یال‌های مابین آنها زیاد بوده اما چگالی یال‌های ارتباطی آنها با دیگر قسمت‌های شبکه کم است. یافتن ساختار انجمنی در گراف، اهمیت زیادی در هر مسئله تحلیل شبکه خصوصاً یافتن الگوهای موجود مابین گره‌های خاصی از شبکه دارد. به همین دلیل، روش‌های فراوانی با رهیافت‌های مختلف برای انجمن‌یابی همپوشان و ناهمپوشان در شبکه‌ها پیشنهاد شده‌اند. البته در سال‌های اخیر، محققین بسیاری برای دسته‌بندی گره‌های گراف از روش‌های یادگیری ویژگی و توکاری شبکه استفاده کرده‌اند. این روش‌ها شبکه را به یک فضای بازنمایی با ابعاد کمتر اما با همان خواص، نگاشت می‌دهند. ما در این مقاله، با الهام از توانمندی شبکه‌های عصبی عمیق در بازنمایی، مدلی را برای یادگیری بازنمایی گراف و نهایتا انجمن‌یابی گره‌های آن پیشنهاد می‌دهیم. در روش پیشنهادی، یک توکاری غیرخطی از گراف توسط کدکننده خودکار پشته‌شده آموزش داده می‌شود و به یک الگوریتم خوشه‌بندی همپوشان اعمال می‌گردد تا انجمن‌های نهایی حاصل شوند. برای نمایش قدرت مدل، آن را به محک‌های استاندارد و چند دادگان با اندازه‌های مختلف از دنیای واقعی اعمال می‌کنیم. نتایج تجربی حاکی از آن است که روش پیشنهادی این مقاله، عملکرد بهتری از چند روش متداول انجمن‌یابی داشته است.

***doi**: 10.5829/ije.2020.33.03c.01*