Saxena et al. *EPJ Data Science* (2022) 11:24
https://doi.org/10.1140/epjds/s13688-022-00336-8

EPJ Data Science
a SpringerOpen Journal

REGULAR ARTICLE Open Access

NodeSim: node similarity based network embedding for diverse link prediction

Akrati Saxena[1*], George Fletcher[1] and Mykola Pechenizkiy[1]

*Correspondence: a.saxena@tue.nl
[1] Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands

Abstract

In real-world complex networks, understanding the dynamics of their evolution has been of great interest to the scientific community. Predicting non-existent but probable links is an essential task of social network analysis as the addition or removal of the links over time leads to the network evolution. In a network, links can be categorized as intra-community links if both end nodes of the link belong to the same community, otherwise inter-community links. The existing link-prediction methods have mainly focused on achieving high accuracy for intra-community link prediction. In this work, we propose a network embedding method, called NodeSim, which captures both similarities between the nodes and the community structure while learning the low-dimensional representation of the network. The embedding is learned using the proposed NodeSim random walk, which efficiently explores the diverse neighborhood while keeping the more similar nodes closer in the context of the node. We verify the efficacy of the proposed embedding method over state-of-the-art methods using diverse link prediction. We propose a machine learning model for link prediction that considers both the nodes' embedding and their community information to predict the link between two given nodes. Extensive experimental results on several real-world networks demonstrate the effectiveness of the proposed method for both inter and intra-community link prediction.

Keywords: Network embedding; Link recommendation; Feature learning

1 Introduction

In online social networks (OSNs), nodes are organized into communities, where a community represents a group of nodes having similar characteristics, such as similar interests, opinions, or beliefs [1, 2]. The links between the nodes belonging to the same community are referred to as *intra community links*, and the links between the nodes belonging to different communities are referred to as *inter community links*. In social networks, intra-community links are driven by the effect of homophily [3] as similar nodes prefer to connect with each other. The formation of inter-community links is still not well explored in the literature; however, it can be explained by different complex phenomena, such as triadic closure and weak ties [4]. In real-world networks, it is observed that the number of intra-community links is more than the number of inter-community links [5]. The evolution of social networks is regulated by the formation of new links in the network.

In OSNs, we recommend more probable, but not existing links as promising connections to help users in making new friends, and a user having more friends will be more loyal towards the website [6, 7]. However, forming the right kind of links is very important as the opinion of a user is highly influenced by the opinion of its neighbors [8]. In the recent era, scientists have focused on increasing the diversity in the network so that the users receive information on a topic from different viewpoints before making their opinion [9]. It is very crucial that a user receives the information from other users having different perspectives to mitigate the negative impact of fake propaganda, false information, or fake news spreading on the network [10]. Hence, it is required that a user has a *diverse* neighborhood by having connections with different communities. In social networks, more inter-community links should be promoted to increase diversity. The link recommendation system plays an important role in forming new links and transforming the network evolution. Besides this, an improved link prediction method can also be used for anomaly detection by better identifying suspicious links in newly formed intra and inter-community links.

Initially, researchers proposed link prediction methods based on the similarity of the nodes [11]. These methods compute the similarity of a pair of nodes based on network structure, and more similar nodes are more likely to form a link. These methods are also often referred to as *classic* or *heuristic link prediction methods*. The well known classic methods include Jaccard coefficient [12], Adamic Adar index [13], resource allocation index [11], preferential attachment index [14], and so on. These methods were extended to include community structure to improve the link prediction accuracy; however, most of the methods improved the total accuracy by improving intra-community link prediction accuracy [15, 16].

In recent works, network characteristics have been studied using network embedding where the network is represented in a low dimensional latent space [17]. In network embedding techniques, the aim is to embed similar nodes closer to each other. Most of the existing network embedding methods [17–19] focus on embedding the nodes closely if they belong to the same community and therefore have high accuracy for the node classification task and intra-community link prediction.

In our work, we propose a network embedding method, called NodeSim embedding, which considers both the nodes' similarity and their community information while generating the network embedding. In the learned embedding, the nodes belonging to the same community will be embedded closely, and the nodes belonging to different communities will be embedded closer based on their neighborhood similarity. Therefore, the generated embedding preserves the structural properties of the network and is efficient in predicting diverse promising links. Next, we propose a link prediction method that trains a logistic regression model using node pair embedding and their community information to predict both the inter and intra-community links with high accuracy. This is the first work that uses community information for learning the link prediction model and achieves higher accuracy for both types of links. The experiments are performed to show the accuracy and efficiency of the proposed method on real-world networks. The results show that the proposed method outperforms the state-of-the-art methods on all the datasets. We further show the application of the proposed method in anomalous link detection, and the NodeSim embedding provides the best results compared to the baseline methods on medium to large-size networks.

The paper is structured as follows. In Sect. 2, we discuss the state of the art literature on link prediction by focusing on network embedding techniques. In Sect. 3, we discuss the proposed methods, including (i) NodeSim network embedding method and (ii) link prediction method. In Sect. 4, we discuss experimental results on real-world networks, including the performance, sensitivity, scalability, robustness analysis, and application of the proposed method. The paper is concluded in Sect. 5 with future directions.

2 Related work

Link prediction is a very well-known problem in network science and has been applied to predict missing links in different types of networks, such as friendship networks, collaboration networks, and chemical networks. Initially, researchers proposed heuristic methods that only considered the neighborhood information of the nodes for link prediction and did not consider the network topology. These heuristic methods were further extended that also considered the network structure properties like community structure to predict the links [15, 16, 20, 21]. However, most of these methods improved the overall accuracy of link prediction by improving the accuracy of intra-community link prediction. The main benefit of using heuristic methods is that these methods do not need any training and are comparatively faster.

Another class of link prediction methods uses machine learning models, such as probabilistic graphical models [22, 23], matrix factorization [24, 25], supervised learning methods [26, 27], and semi-supervised learning methods [28, 29]. These machine learning methods provide good accuracy though they suffer from the class imbalance problem as the number of existing links in a network are significantly fewer than the number of non-existing links.

In recent years, network embedding techniques have been used to study networks and to propose solutions for various network analysis problems. The network embedding methods can be categorized into three categories based on the structural proximity considered while generating the embedding, (i) microscopic structure embedding, which considers local proximity of nodes, such as first-order [30, 31], second-order [30] or high-order proximity [17, 18, 32], (ii) mesoscopic structure embedding, which captures hierarchical and community structural proximity [33–35], and (iii) network properties preserved embedding, which captures global network properties, such as network transitivity or structural balance [36, 37].

In the existing mesoscopic network embedding, the main focus has been either on the hierarchical embedding where the users belonging to the same hierarchy should be embedded together [33] or on the intra-community proximity where the nodes belonging to one community should be embedded closely [34, 35]. In hierarchical or structural role proximity, the nodes playing the same structural roles are embedded closely; for example, the nodes having a similar degree or similar influential power should be embedded closer [38–43]. In this work, we propose the NodeSim network embedding method, which considers both (i) high-order proximity by the similarity of the nodes and (ii) mesoscopic structure by the network communities while generating the embedding. In NodeSim embedding, the nodes belonging to one community are clustered together, and the similar nodes belonging to different communities are embedded closer. The proposed embedding captures a richer diverse neighborhood of the nodes that is further verified using the link prediction.

3 The proposed method

In this section, we first discuss the required network properties for our work. Next, we discuss our proposed NodeSim embedding method to learn the feature representation of the nodes and the proposed link prediction method.

3.1 Community structure

In real-world complex networks, nodes connect with each other if they have similar properties. A group of nodes that are densely connected with each other is referred to as a community [44]. The community label of a node u is denoted by C_u. If both end nodes of a link (u, v) belong to the same community, it is referred to as an intra-community link, and $C_{(u,v)} = 1$ for an intra-community link. If both end nodes belong to different communities, then the link (u, v) is referred as an inter-community link and $C_{(u,v)} = 0$.

In most real-world networks, the ground truth community information is not available. In literature, several community detection methods have been proposed to identify communities using network structure if the ground truth information is not known. In this work, we apply the highly used community detection method, known as the Louvain method [45], to identify the communities if the ground truth information is not known.

Louvain Community Detection Method: The Louvain method [45] uses two-step greedy optimization to optimize the modularity of a community partition of the network. First, the method optimizes the modularity locally to find small communities. In the second step, it merges all nodes belonging to the same community and creates an aggregated network where each node represents a community. These steps are performed iteratively until we achieve the maximum modularity and the obtained communities are returned.

3.2 Node-pair similarity

In a network, two nodes connect with each other if they have some common interest or characteristics, and therefore, a link between a pair of nodes is the first indication that they are similar. However, these binary/unweighted connections cannot capture the complete information of the system as each connection is not equally important. A better way of representing the network is with weighted edges, where edge-weight denotes the strength of the connection. For example, in a friendship network, the weight of an edge can be computed based on the intimacy of the relationship or frequency of the communication [46]. The similarity of a node pair (u, v) is denoted as $\mathrm{Sim}(u, v)$.

In most real-world networks, the edge-weight data is not available as it is not feasible to collect all the required information for computing the strength of each connection. In network science, there have been proposed methods to compute the similarity of a node-pair based on their neighborhood connectivity in the network structure. Some of the well-known methods are the number of common neighbors [47], Jaccard coefficient [12], Adamic Adar [13], Resource Allocation [11], and so on, which compute a node-pair similarity based on their local-neighborhood proximity.

In this work, we will use the Jaccard coefficient to compute a node pair's similarity in unweighted networks. The Jaccard coefficient for a node pair (u, v) is defined as, $\mathrm{JC}(u, v) = \frac{|\Gamma(u)\cap\Gamma(v)|}{|\Gamma(u)\cup\Gamma(v)|}$, where $\Gamma(u)$ is the set of neighbors of node u.

3.3 NodeSim network embedding

For a given graph $G(V, E)$, the network embedding method learns the mapping $\Phi : V \rightarrow \mathbb{R}^d$, where d is the dimension of the embedding space. In recent works, the Skip-gram model has been used to generate the network embedding by representing the network as a document where the nodes are corresponding to the words [18]. In a network, a sampled sequence of nodes is considered the same as an ordered sequence of words in a document. The simplest way to generate the ordered sequence of nodes is by using random walks.

In the random walk [48], if the random walker is at node u, the probability that the random walker will move to node v is defined as,

$$P_{uv} = \begin{cases} 1/\deg(u), & \text{if } (u,v) \in E, \\ 0, & \text{otherwise.} \end{cases}$$

The random walk method does not consider the network structure properties while sampling the nodes. In recent works, different sampling methodologies have been explored to sample the network to learn feature representations of the network [17, 49]. However, the proposed methods do not consider the meso-scale properties, such as community structure, while exploring the network. In this work, we propose a random walk based sampling method, called NodeSim Random Walk, that captures the neighborhood of the node by considering both the nodes' similarity as well as the meso-scale community structure of the network.

3.3.1 NodeSim random walk

In network embedding, the focus is to embed similar nodes closer. The simplest way to capture the node similarity during the random walk would be to bias the edge probability based on the similarity of its end nodes. However, this will ignore the meso-scale property of the network that is captured through the community structure. In NodeSim random walk, the edge-probabilities are assigned based on both the similarity of the nodes and community structure.

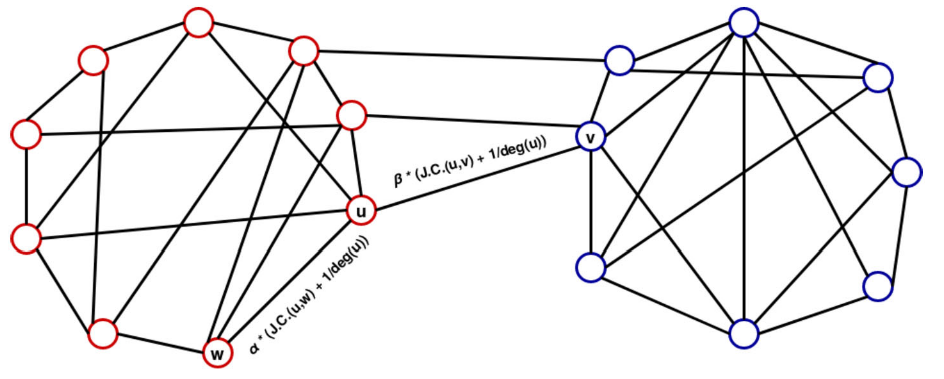
In NodeSim Random walk, the unnormalized probability p_{uv} to move from node u to node v is defined as,

$$p_{uv} = \begin{cases} \alpha \cdot (\mathrm{Sim}(u,v) + 1/\deg(u)), & \text{if } (u,v) \in E \text{ and } C_{(u,v)} = 1, \\ \beta \cdot (\mathrm{Sim}(u,v) + 1/\deg(u)), & \text{if } (u,v) \in E \text{ and } C_{(u,v)} = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The probabilities are normalized for each node u with respect to all of its neighbors. So, the probability to move from node u to node v is computed as, $P_{uv} = p_{uv} \cdot w_u$ where w_u is the normalizing factor for node u.

In this work, the similarity of the nodes is computed using the Jaccard Coefficient. Figure 1 explains edge-probabilities for NodeSim random walk, where the network has two communities shown by red and blue nodes, and the edges (u,v) and (u,w) are inter and intra-community edges, respectively, which are labeled with p_{uv} and p_{uw}, respectively.

Intuitively, parameters α and β control how the random walker explores the neighborhood. A higher value of α shows that the walker will prefer to sample more similar nodes from the same community, and a higher value of β shows that the walker will put a higher weight to explore the inter-community neighborhood of the node.



v
u
w
β * (J.C.(u,v) + 1/deg(u))
α * (J.C.(u,w) + 1/deg(u))

Figure 1 NodeSim Random Walk probabilities for inter and intra community nodes

3.3.2 Learn embedding

Once the ordered sequences of nodes are generated using NodeSim random walk, the network embedding is learned using the Skip-gram model [50]. The network embedding method learns a mapping for each node $u \in V$ to a d-dimension embedding space that represents the d-dimensional feature representation of node u based on its structural role. The network embedding is denoted as $\Phi : u \in V \longrightarrow \mathbb{R}^{|V|\times d}$, where Φ can be considered a $|V| \times d$ size matrix that is learned by solving a maximal likelihood optimization problem.

In the skip-gram model, given the corpus, the neighborhood of a word is defined using a sliding window over the consecutive words. In networks, we generate the ordered sequence of nodes using sampling methods. For example, if NodeSim random walker visits the following nodes $\{u_1, u_2, \dots u_i, \dots u_l\}$, they will be referred to as an ordered sequence of nodes. In our method, we generate ordered sequences of nodes by taking γ NodeSim walks of length l from each node. The neighborhood of a node u_i will be defined by considering $k - 1$ nodes visited before and after node u_i during the sampling, where k is the window size or context of the node. For every node $u_i \in V$, $N_{NS}(u_i) \subset V$ denotes the neighborhood of node u_i in the network that is generated through the NodeSim sampling method with the given context k.

In the skip-gram model, the network embedding is learned based on the likelihood of a node u_i co-occurring with other neighborhood nodes within the context k in the NodeSim random walk. We, therefore, optimize the following optimization function that aims for maximizing the probability of observing a node in the neighborhood of node u_i, given its feature representation $\Phi(u_i)$,

$$\underset{\Phi}{\text{maximize}} \sum_{u_i \in V} \log \Pr\bigl(N_{NS}(u_i) | \Phi(u_i)\bigr). \tag{2}$$

The optimization problem is solved using two assumptions. The first assumption is conditional independence, that the probability of observing a node in the neighborhood of the source node is independent of observing any other node in its neighborhood given the feature representation of the source node, so,

$$\Pr\bigl(N_{NS}(u_i) | \Phi(u_i)\bigr) = \Pi_{u_j \in N_{NS}(u_i)} \Pr\bigl(u_j | \Phi(u_i)\bigr). \tag{3}$$

The second assumption is the symmetry that considers the pairwise similarity of a source node and its neighborhood node in the feature space. Therefore, we estimate the probability of a node u_j co-occurring with node u_i using the softmax function,

$$\Pr\bigl(u_j|\Phi(u_i)\bigr) = \frac{\exp(\Phi(u_j)\cdot\Phi(u_i))}{\sum_{v\in V}\exp(\Phi(v)\cdot\Phi(u_i))}. \tag{4}$$

Finally, using both assumptions, the objective function given in Equation (2) is computed as,

$$\underset{\Phi}{\text{maximize}} \sum_{u_i\in V}\Biggl(-\log Z_{u_i} + \sum_{u_j\in N_{NS}(u_i)} \Phi(u_j)\cdot\Phi(u_i)\Biggr), \tag{5}$$

where $Z_{u_i} = \sum_{v\in V}\exp(\Phi(u_i)\cdot\Phi(v))$ is expensive for large-scale networks and it is approximated using negative sampling method [51]. Equation (5) is optimized using SGA (stochastic gradient ascent) over the features Φ [17].

3.3.3 Complexity

The complexity of the proposed network embedding method depends on two major steps, (i) identify the communities and (ii) NodeSim embedding learned using the Skip-gram model. The complexity of the community detection method and Skip-gram model is well defined in the literature, so we briefly discuss the complexity of our method. In our implementation, we have used the Louvain community detection method having complexity $O(n\cdot\log n)$ where n is the total number of nodes in the network. Once the community structure is identified, the complexity to generate the probability distribution for NodeSim random walk is $O(m)$ where m is the total number of edges in the network. The complexity for learning embedding using the skip-gram model is $O(nkl\gamma(d + d\log(n)))$, where d denotes the number of dimensions, l denotes the walk length, k denotes the window size, and γ denotes the number of random walks. So, the overall complexity is $O(n\log n + m + nkl\gamma(d + d\log(n)))$.

3.4 Link-prediction method

The link prediction method first generates the feature representation of given node pairs and then train a logistic regression model using the feature representation of node pairs and their community information.

3.4.1 Feature representation of node pair

The feature representation of a pair of node (u,v) is generated by applying a binary operator on the feature representation of node u and v. The most common operators are mentioned below.

1. Average: $e_i(u,v) = \frac{\Phi_i(u)+\Phi_i(v)}{2}$
2. Weighted-L1: $e_i(u,v) = |\Phi_i(u) - \Phi_i(v)|$
3. Weighted-L2: $e_i(u,v) = |\Phi_i(u) - \Phi_i(v)|^2$
4. Hadamard: $e_i(u,v) = \Phi_i(u) * \Phi_i(v)$

$\Phi_i(u)$ denotes the i_{th} feature of node u, and $e_i(u,v)$ denotes the i_{th} feature of a node pair (u,v). In this way, a d-dimension feature vector is generated for each node-pair using the d-dimension feature representation of the corresponding nodes.

3.4.2 Link prediction model

For link prediction, a logistic regression model is trained using features of the node-pair and their community information, with the output having the existent/non-existent information of the link between the given node-pair. The input features for a node pair (u, v) is generated as, $f(u, v) = (e(u, v)||C_{(u,v)})$, where $||$ is concatenation operator and $C_{(u,v)}$ is 1 if both nodes u and v belong to the same community, otherwise 0. The output parameter is 1 or 0 if there exists a link between the given pair of nodes or not, respectively. We have shown results for all four operators applied on $e(u, v)$.

4 Experimental analysis

In this section, we discuss baseline methods, datasets, and experimental results.

4.1 Baseline methods

The proposed method is compared with both types of link prediction methods (i) similarity-based heuristic methods and (ii) network embedding based methods.

We compare with the following three heuristic methods based on network structure.

1. Jaccard Coefficient (JC) [12]: $\mathrm{JC}(u, v) = \frac{|\Gamma(u)\cap\Gamma(v)|}{|\Gamma(u)\cup\Gamma(v)|}$
2. Adamic Adar (AA) [13]: $AA(u, v) = \sum_{w\in(\Gamma(u)\cap\Gamma(v))} \frac{1}{\log|\Gamma(w)|}$
3. Resource Allocation (RA) [11]: $RA(u, v) = \sum_{w\in(\Gamma(u)\cap\Gamma(v))} \frac{1}{|\Gamma(w)|}$

We compare our method with the following network embedding based link-prediction methods.

4. DeepWalk [18]: Deepwalk method learns the network embedding using the skip-gram model on the ordered sequence of nodes generated using random walk.
5. Node2Vec [17]: Node2Vec is an extension of DeepWalk where the walker has different probabilities for moving to its neighbors, and the probability to move to the next node depends on its distance from the previously visited node. Once the nodes are sampled, the network embedding is learned using the skip-gram model. We have used the code provided by the authors at https://github.com/aditya-grover/node2vec.
6. NECS [35]: Network Embedding with Community Structural information (NECS) uses nonnegative matrix factorization to generate nodes' embedding, which preserves the high-order proximity. The final network embedding is learned by jointly optimizing the consensus relationship between the nodes' representation and the community structure. We have used the implementation provided by the authors at https://github.com/liyu1990/necs.

   For DeepWalk, Node2Vec, and NECS methods, the node-pair embedding is generated using the Hadamard operator, and then the logistic regression model is trained for the link prediction as mentioned in these works.
7. Splitter [19]: This network embedding method learns multiple embedding of each node based on the principled decomposition of the ego-network. These multiple representations of a node denote its embedding with respect to the local communities it belongs to. The implementation is provided by the authors at https://github.com/google-research/google-research/tree/master/graph_embedding/persona. For link prediction, we used the method discussed in their paper. For each node pair (u, v), the similarity score is computed using the dot product of their embedding. In the persona graph, each node has multiple embedding, so we compute the similarity score

Table 1 Datasets

Network	#nodes	#edges	#communities	Ref
Facebook	4039	88,234	16	[52]
GrQc	4158	13,422	42	[53]
Hep-th	8638	24,806	50	[53]
Hep-ph	11,204	117,619	38	[53]
Astro-ph	17,903	196,972	37	[53]
Enron	33,696	180,811	178	[54]
Twitter	81,306	1,342,296	77	[52]
DBLP	317,080	1,049,866	216	[55]

for each combination of their embedding, and the maximum score is returned as the final similarity score.

The implementation code of NodeSim method is available at https://github.com/akratiiet/NodeSim.

4.2 Datasets

We perform experiments on real-world networks, and their details are mentioned in Table 1. Facebook and Twitter are snapshots from online social networking websites, Enron is an email communication network, and GrQc, Hep-th, Hep-ph, Astro-ph, and DBLP are co-authorship networks. In all the networks, the communities are detected using the Louvain Method, and a community label is assigned to each node based on which community it belongs to. A node pair is referred to as intra-community node pair if both the nodes belong to the same community; otherwise, it will be referred to as inter-community node pair.

To generate the training and testing data, we follow the same methodology as used in [17, 19]; however, we maintain the ratio of inter and intra-community links that is not considered in previous studies. First, we remove 10% of inter-community and 10% of intra-community edges from E uniformly at random and put them in set E_{lp} that will be used for link prediction. While removing the 10% edges, it is ensured that the network remains connected. The remaining 90% edges are referred to as E_{ne}, and $G(E_{ne}, V)$ will be used to generate network embedding.

For link prediction task, the same number of inter and intra-community node pairs for non-existent links are chosen uniformly at random, as we have in E_{lp}. These sampled links will work as negative cases and are added to set E_{lp}. If a link is formed between a given node pair, then it is referred to as a positive case; otherwise, it will be referred to as a negative case. To create train and test data, the node pairs in E_{lp} are split into E_{train} and E_{test}, and while splitting, we ensure that the ratio of intra and inter-community node pairs is maintained for both positive and negative cases. The default train and test ratio is $(0.5 : 0.5)$ if it is not mentioned explicitly. In heuristic and Splitter link Prediction methods, a node pair is predicted positive if the similarity score for this pair is higher than the similarity score of 50% positive train cases.

4.3 Performance study

First, we compare NodeSim method with baselines, and ROC-AUC value is computed for all test cases, intra-community and inter-community test cases as shown in Table 2. The table shows the best results observed for different parameter settings used in different methods, and each experiment is repeated five times to compute the average. The dimension

Table 2 ROC-AUC for link prediction

Method/Datasets		Facebook	GrQc	Hep-th	Hep-ph	Astro-ph	Enron	Twitter	DBLP
JC	Total	0.713	0.756	0.735	0.741	0.750	0.731	0.743	0.750
	Intra	0.721	0.788	0.782	0.801	0.827	0.709	0.774	0.749
	Inter	0.5	0.5	0.541	0.502	0.578	0.752	0.527	0.751
AA	Total	0.747	0.766	0.752	0.746	0.749	0.746	0.748	0.754
	Intra	0.751	0.790	0.785	0.772	0.796	0.745	0.767	0.755
	Inter	0.642	0.568	0.617	0.643	0.641	0.746	0.612	0.753
RA	Total	0.751	0.770	0.752	0.744	0.751	0.747	0.747	0.752
	Intra	0.754	0.795	0.782	0.786	0.799	0.748	0.765	0.753
	Inter	0.677	0.568	0.625	0.575	0.643	0.745	0.627	0.751
DeepWalk	Total	0.796	0.793	0.797	0.886	0.838	0.762	0.842	0.862
	Intra	0.804	0.819	0.832	0.906	0.876	0.721	0.858	0.831
	Inter	0.587	0.582	0.652	0.807	0.754	0.804	0.735	0.893
Node2Vec	Total	0.829	0.820	0.830	0.900	0.865	0.769	0.860	0.883
	Intra	0.834	0.849	0.864	0.916	0.897	0.731	0.877	0.854
	Inter	0.619	0.582	0.691	0.834	0.793	0.806	0.742	0.911
Splitter	Total	0.752	0.806	0.729	0.865	0.847	0.746	*	*
	Intra	0.765	0.823	0.733	0.902	0.905	0.744	*	*
	Inter	0.433	0.666	0.712	0.718	0.718	0.748	*	*
NECS	Total	0.548	0.544	0.549	0.581	*	*	*	*
	Intra	0.550	0.546	0.553	0.581	*	*	*	*
	Inter	0.509	0.527	0.533	0.580	*	*	*	*
NodeSim (Average)	Total	0.758	0.836	0.596	0.837	0.690	0.818	0.696	0.717
	Intra	0.758	0.838	0.591	0.839	0.694	0.777	0.722	0.702
	Inter	0.756	0.829	0.617	0.826	0.681	0.858	0.513	0.732
NodeSim (Wtd.-L1)	Total	0.824	0.816	0.784	0.876	0.829	0.676	0.808	0.658
	Intra	0.833	0.845	0.827	0.911	0.866	0.619	0.831	0.632
	Inter	0.581	0.582	0.607	0.740	0.747	0.733	0.650	0.683
NodeSim (Wtd.-L2)	Total	0.827	0.833	0.783	0.875	0.831	0.691	0.808	0.654
	Intra	0.834	0.863	0.820	0.908	0.864	0.632	0.828	0.628
	Inter	0.651	0.589	0.628	0.745	0.758	0.750	0.667	0.680
NodeSim (Hadamard)	Total	**0.857**	**0.864**	**0.849**	**0.924**	**0.883**	**0.835**	**0.901**	**0.904**
	Intra	**0.862**	**0.874**	**0.884**	**0.937**	**0.907**	**0.794**	**0.914**	**0.884**
	Inter	**0.736**	**0.706**	**0.749**	**0.872**	**0.828**	**0.877**	**0.809**	**0.924**

of network embedding is $d = 128$. The results show that the proposed NodeSim method with Hadamard operator for node pair embedding outperforms all baseline methods. The bold faced values show the best ROC-AUC obtained for the total link prediction and the best results also provide better Intra and Inter link prediction results as compared to all the baselines. The '*' value for NECS and Splitter methods show that the code execution was not completed in 48 hours on the server, and therefore, the values are not mentioned. The NECS method uses matrix factorization and therefore has high computational complexity. The Splitter method generates multiple embedding of each node corresponding to its local communities, and therefore, the execution time is manyfold based on the density of the network and connectivity of the nodes.

We further study the performance of our method by varying the ratio of train and test set. The results are shown in Fig. 2 for Hep-ph and Astro-ph networks. Results show that the performance of the proposed method is better compared to baselines, even if the training ratio is 0.1; however, the best results are achieved when the ratio of training size is at least 0.5 and 0.3 for Hep-ph and Astro-ph networks, respectively.

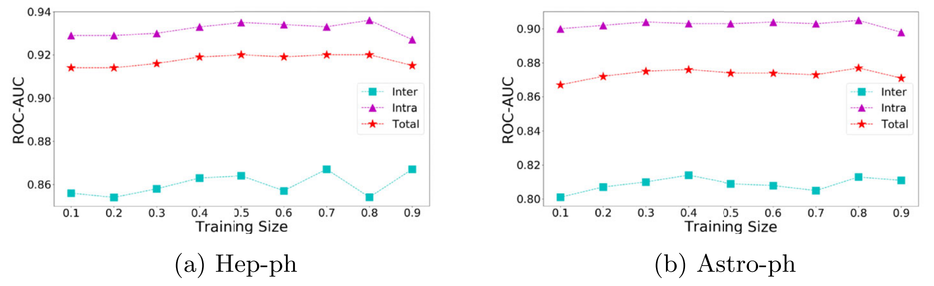


Figure 2 Vary train size

Table 3 Default and varied range values of different network embedding parameters

Parameter	Default	Range
α	1	1, 1.5, 2, 2.5, 3
β	1.5	1, 1.5, 2, 2.5, 3
Dimension (*d*)	128	4, 8, 16, 32, 64, 128, 256
Context (*k*)	5	5, 7, 9, 11, 13, 15
Number of Walks (γ)	10	6, 8, 10, 12, 14, 16, 18, 20
Walk Length (*l*)	80	40, 50, 60, 70, 80, 90, 100

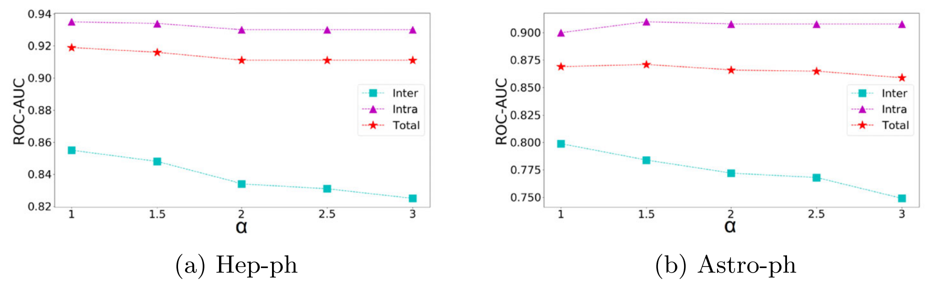


Figure 3 Impact of varying α

4.4 Parameter sensitivity

The NodeSim embedding method depends on a number of parameters, and we examine the impact of different parameters on the performance of link prediction. In Table 3, we have shown the default values of different parameters that has been decided based on the preliminary analysis and their range that we have considered. The results are shown on two networks, Hep-ph and Astro-ph.

Figure 3 shows the impact of varying α on inter and intra-community link prediction. The results show that $\alpha \sim 1$–1.5 achieves the best results. In Fig. 4, the results show that $\beta \sim 1.5$–2 achieves the best results. The results confirm that the inter-community edges should be weighted higher than the intra-community edges during the sampling to predict inter-community links with high accuracy, as expected.

Next, we analyze the impact of embedding parameters on link prediction accuracy. Figure 5 represents that the performance of link prediction methods improves with the embedding dimension. In Fig. 6, we observe that the performance reduces with the window size as the larger window size considers distant nodes while generating the local context of the nodes, and these nodes might not be similar. In real-world networks, most of the

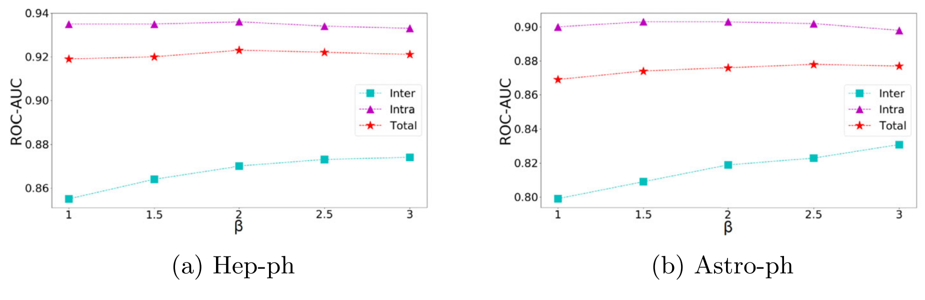


Figure 4 Impact of varying β

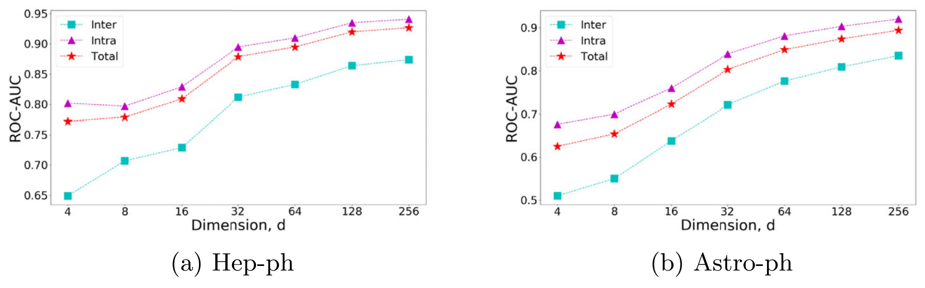


Figure 5 Impact of varying Dimension (d)

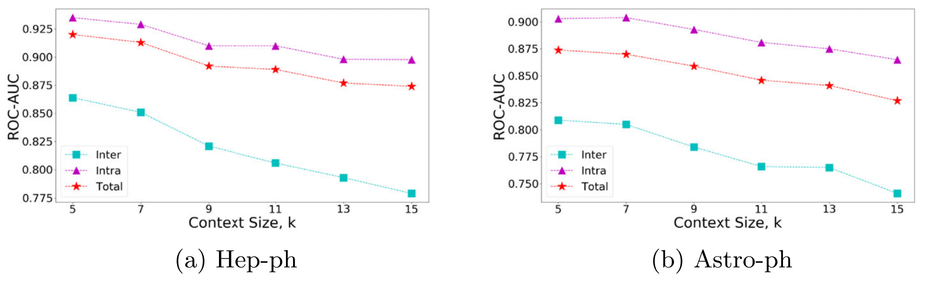


Figure 6 Impact of varying context k

new links are driven by the triad closure phenomenon, and it is less probable that a node will be connected to a distant node.

Figures 7 and 8 show results for varying the number of walks and the walk-length. As observed in Fig. 7, the intra-community results are less affected by the number of walks than the inter-community links as the ratio of inter-community context pairs decreases with more number of walks; as we expected. Similarly, the inter-community accuracy also decreases with the walk-length even if the total accuracy is improved, as shown in Fig. 8 (b).

4.5 Scalability

We compare the running time of different network embedding based methods on synthetic networks generated using SCCP (Scale-free networks with Community and Core-Periphery) model [56, 57]. The network generator first creates a seed graph, i.e., a complete

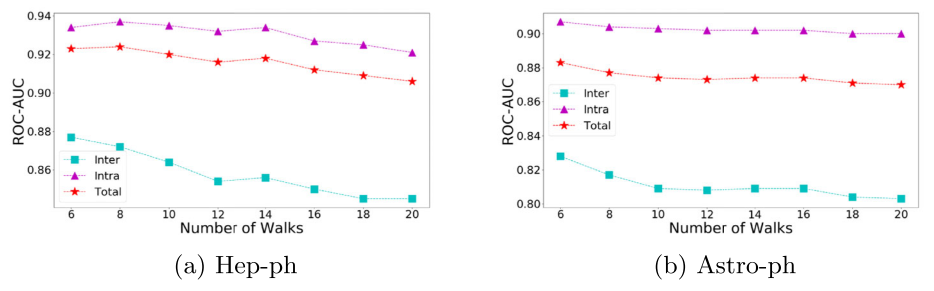


Figure 7 Impact of varying Number of Walks

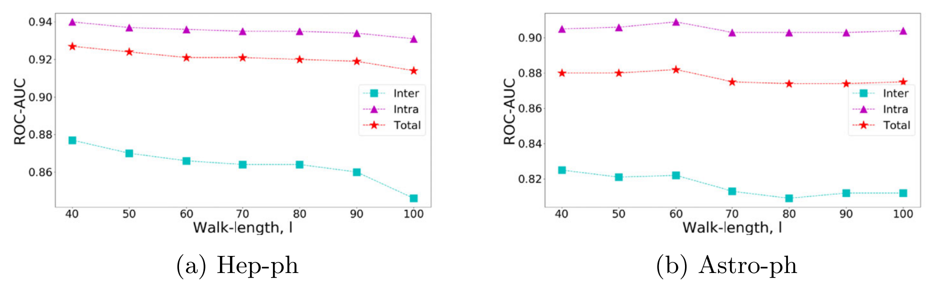


Figure 8 Impact of varying Walk-length

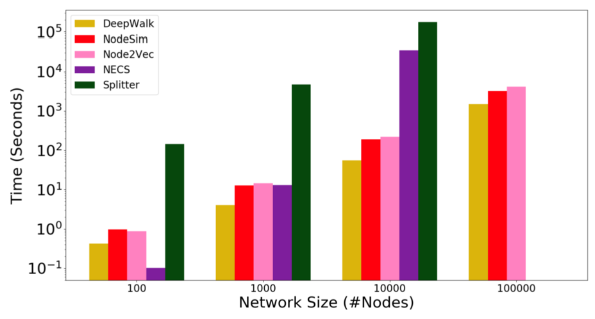


Figure 9 Running time for different embedding methods versus network size

graph of m nodes for each community, where m is the average degree of nodes. Next, in each iteration, a new node is added to each community, and the added node builds m connections using preferential attachment law [14] while ensuring the intra and inter-community edge ratio. The running time is compared on synthetic networks so that the ratio of intra and inter-community edges are maintained as we increase the network size. In our experiments, the ratio is (intra : inter = 0.75 : 0.25), and the average degree of the network is 8. The total number of communities is 10 in the network having 100 and 1000 nodes and 100 in the network having 10,000 and 100,000 nodes. All communities in a network are of the same size.

Figure 9 show the running time of different methods. All experiments are performed on the server having 384 GB RAM and 2× Intel Xeon 4110 @ 2.1 GHz CPU. For 100,000 nodes network, the Splitter code was not finished in 48 hours, and the NECS code was

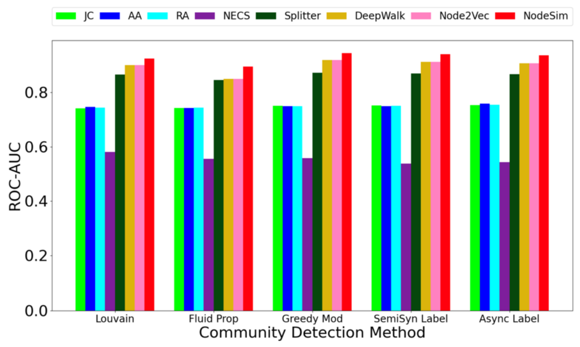


Figure 10 ROC-AUC for link prediction corresponding to different community detection methods for Hep-ph network

killed due to the memory error on the server. The results show that the proposed method executes faster than all the baselines except deepwalk as the network size grows. The running time of NodeSim and Node2Vec is almost equal. The deepwalk method is the fastest as it creates node context using a simple random walk and does not consider the structural properties of the network.

4.6 Robustness for identified communities

There have been proposed several community detection methods in the literature that consider different network properties while identifying the communities. Therefore, the communities identified by different methods might vary. For some methods, such as Louvain or greedy method, if the same method is applied many times, the returned community structure might differ each time.

We, therefore, study the efficiency of the NodeSim embedding method corresponding to different community detection methods. We apply five different community detection methods (including Louvain): (i) Louvain method [45]. (ii) Fluid Communities Algorithm [58], (iii) Greedy Modularity Maximization [2], (iv) Semi-synchronous Label Propagation [59], and (v) Asynchronous Label Propagation [60]. The details of community detection methods are explained in Appendix A. After identifying the communities using different methods, the training and testing data is created, as discussed in Sect. 4.2. Next, we generate network embedding by applying different embedding methods and apply the link prediction method. Each method is executed five times, and the average ROC-AUC value for the Hep-ph network is shown in Fig. 10.

The results show that the performance of different methods is relatively maintained irrespective of the community detection method. The NodeSim method outperforms in all the cases as the method considers both the similarity of nodes and their communities while generating the network embedding.

4.7 Case study

For visualization, we show the NodeSim embedding of the Zachary Karate Network [61] in 2-dimension and 3-dimension space. The network and its embeddings are shown in Fig. 11, where the nodes having the same color belong to one community. The embedding shows that the nodes belonging to different communities are well separated; however,

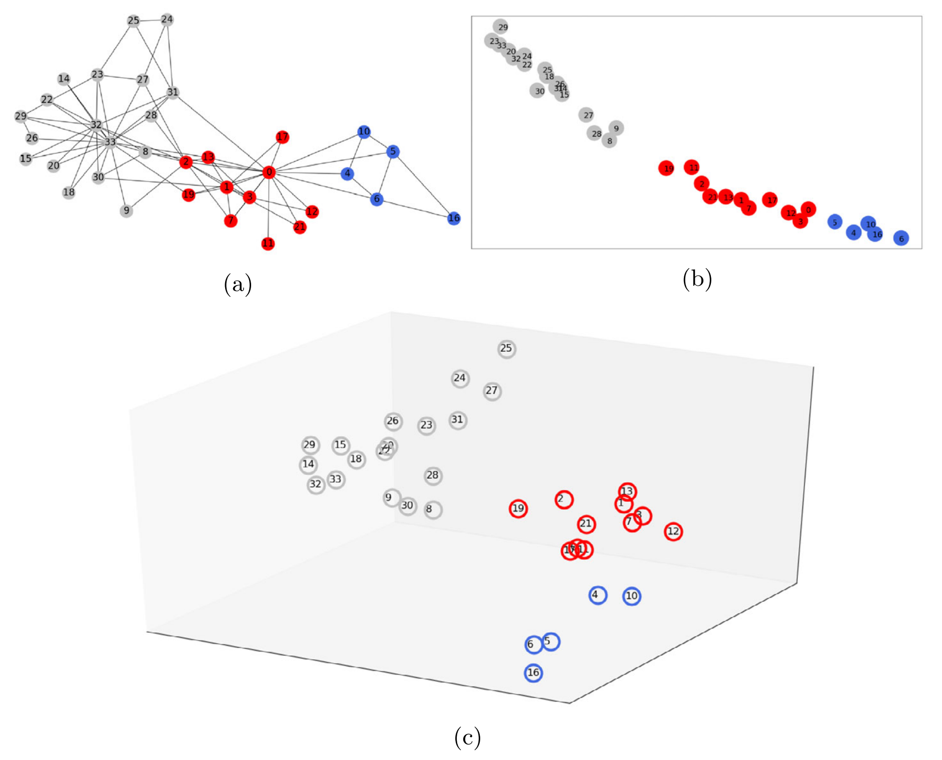


Figure 11 (**a**) Zachary Karate Network with three communities, (**b**) and (**c**) 2-dimensional and 3-dimensional embedding of Zachary Karate Network using NodeSim Method, respectively

more similar nodes are embedded closer. For example, node 12 is more likely to form inter-community links with nodes 4, 5, 6, and 10, so, as observed, they are embedded closer but still well separated. The embedding of the nodes improves with high dimension, as we also observed in Sect. 4.4 that the accuracy increases with a higher dimension. We have also shown embeddings for Dutch School Friendship Network [62], and Illinois Highschool Friendship Network [63] in Appendix B.

4.8 Application in anomaly detection

The one well-known application of link prediction is to detect anomalous links. We briefly analyze the performance of the proposed method for anomalous link prediction. For this analysis, we use four real-world anomaly datasets and three synthetic datasets generated using real-world network; the details are provided in Table 4. The German Boys network [64] is a friendship network of a German school class from 1880–1881, and students are labeled as outliers based on their characteristics and behavior. The Disney and Books networks are co-purchase networks extracted from Amazon [65]. Enron-Anomaly [66] is an email communication network having spammers labeled as anomalous users. In all real-world anomaly datasets, nodes are labeled as anomalous and non-anomalous. To create synthetic anomaly network datasets, we follow the method used in previous anomaly detection works [67, 68]. We first add 0.4% nodes as anomalous nodes to the given network G. Each anomalous node picks its degree (k) from the degree-distribution of network G and will make k connections uniformly at random from the nodes of network G. The synthetic anomalous networks corresponding to *Hep-ph, Astro-ph, and DBLP* networks

Table 4 Datasets for anomaly analysis

Datasets	#Nodes	#Edges	#Anomalous Edges	#Regular Edges	Ref
German Boys	48	149	82	67	[64]
Disney	124	335	17	318	[65]
Books	1418	3695	229	3466	[65]
Enron-Anomaly	13,533	176,987	28	176,959	[66]
Hep-ph Anomaly	11,248	118,969	1350	117,619	[53]
Astro-ph Anomaly	17,974	198,346	1374	196,972	[53]
DBLP Anomaly	318,348	1,058,175	8309	1,049,866	[55]

Table 5 ROC-AUC for anomalous link detection

Methods/Datasets	German-Boys	Disney	Books	Enron Anomaly	Hep-ph Anomaly	Astro-ph Anomaly	DBLP Anomaly
DeepWalk	0.741	0.799	0.941	0.884	0.906	0.789	0.647
Node2Vec	0.635	0.793	0.941	0.884	0.905	0.800	0.658
NECS	**0.864**	0.768	0.796	0.877	0.800	*	*
NodeSim	0.606	**0.865**	**0.944**	**0.960**	**0.932**	**0.835**	**0.774**

Table 6 Micro-F1 for anomalous link detection

Methods/Datasets	German-Boys	Disney	Books	Enron Anomaly	Hep-ph Anomaly	Astro-ph Anomaly	DBLP Anomaly
DeepWalk	0.740	0.843	0.927	**0.999**	0.935	0.831	0.668
Node2Vec	0.644	0.831	0.927	**0.999**	0.932	0.846	0.681
NECS	**0.863**	**0.898**	0.909	0.907	0.637	*	*
NodeSim	0.603	0.855	**0.941**	0.998	**0.939**	**0.876**	**0.801**

are referred to as *Hep-ph Anomaly, Astro-ph Anomaly, and DBLP Anomaly*, respectively. In all networks, each edge that is connected with any anomalous node is labeled as an anomalous edge, and the rest of the edges are considered regular edges (also referred to as non-anomalous edges).

For anomalous like detection, we create network embedding using Deepwalk, Node2Vec, NECS, and NodeSim methods. For German Boys school, we create 32 dimension embedding as it is a small network, using the following parameters: 5 number of walks of length 10, 3 window size, $p = 0.25$ and $q = 0.25$ for Node2Vec, and $\alpha = 1$ and $\beta = 1$ for NodeSim embedding method. For other networks, we create 128 dimension embedding using default parameter settings for the number of walks, walk-length, and window size as used for link prediction. Please note that the Splitter method generates multiple embedding of each node based on its local persona, and therefore, this can not be used directly for anomaly detection.

To create train and testing data, we uniformly split 50% anomalous and non-anomalous edges as training dataset and the rest 50% as testing data. Given that the training data is imbalanced due to a very small number of anomalous edges, we use SMOTE (Synthetic Minority Oversampling Technique) oversampling [69] to create a balanced dataset using two nearest neighbors. Then, we train a logistic regression model on the balanced dataset for different network embeddings. The ROC-AUC and Micro-F1 values for the testing data are shown in Table 5 and 6, respectively. We observe that for very small networks, such as German boys, the NECS method provides better results. However, the performance of NECS is the worst for medium-size networks, and the method has a very high computational complexity for large-size networks. The results show that the NodeSim method provides promising results for anomalous link detection for medium to large-scale networks.

We have shown detailed experiments for anomalous link detection as this is a more suitable application of link prediction; however the NodeSim embedding can also be used for detecting anomalous nodes. For anomalous nodes detection, one can use any of the following two approaches, (i) directly train a machine learning model on the network embedding to identify anomalous nodes, or (ii) first classify the edges as anomalous and non-anomalous, and then use this information further to classify anomalous nodes. One of the main limitations in anomalous link prediction is the availability of real-world datasets. In our analysis, we consider that each link connected to an anomalous node is anomalous; however, in real-world applications, there might be a case where an anomalous node can have both types of connections, anomalous as well non-anomalous. In the proposed method, we only use network structure, and the method provides good results compared to baselines with the limited information of the network where no additional information is available due to the privacy concerns of the users. Given the promising performance of the NodeSim method, one can use it further for designing improved anomalous links and nodes detection methods using the additional information of the nodes and network. For example, in attributed networks, the anomaly link detection method can use network embedding and nodes' attributes to achieve improved performance.

5 Conclusion

In this work, we have proposed the NodeSim network embedding method, which considers both the nodes' similarity and their community membership while learning the feature representation of the nodes. The NodeSim embedding method efficiently learns the embedding of diverse nodes that is further verified using the link prediction. We proposed a link prediction method that trains a logistic regression model using nodes' features and their community information. The results showed that the proposed link-prediction method outperforms baseline methods for both intra-community as well as inter-community link prediction. We further studied the impact of different parameters and showed that a higher value of β provides higher inter-community link prediction accuracy as the NodeSim method embeds the more similar diverse nodes closer than the others. We further show the application of the proposed method in anomaly detection and network visualization.

In the future, we would like to extend the proposed method to generate embedding of dynamic networks to predict inter and intra-community links with high accuracy to increase diversity. Such embedding can be used for several downstream tasks in dynamic networks, such as anomaly detection, network visualization, and recommendation systems for suggesting content, posts, or advertisements.

Appendix A: Community detection methods

Here we discuss the details of community detection methods other than the Louvain method that we have applied for studying the robustness of NodeSim network embedding.

1. Asynchronous Label Propagation [60]: In this method, each node is initialized with a unique community label. In every iteration, each node will update its community

label based on its neighbors' community label at that time. Thus, the nodes belonging to a strongly connected group will be assigned the same community label with their consensus through this iterative process.

2. Semi-synchronous Label Propagation [59]: This method is similar to the asynchronous Label propagation method, and it combines the advantages of both synchronous and asynchronous method. In this method, each node is assigned with a community label initially, and at each iteration, a node updates its community label based on the most used label by its neighbors. However, the ties are broken randomly, and the method is stopped when no node changes its label.
3. Fluid Communities Algorithm [58]: Fluid communities are based on the idea of fluids interacting with each other, such as expanding or pushing each other in an environment. In this method, first, each of the initial *c* communities is initialized by a random node in the network. Then, in each iteration, each node's community label is updated based on its community and the community of its neighbors. Once no node changes its community in an iteration, the method is stopped. In our implementation, we set the number of communities approximately close to the number of communities identified by the Louvain method.
4. Greedy Modularity Maximization [2]: This method is a well-known method to identify communities by maximizing the modularity in the network. In this method, each node is assigned with a community label, and in each step, two communities are merged that most increases the modularity. The method is stopped when the modularity can not be further increased by merging two communities.

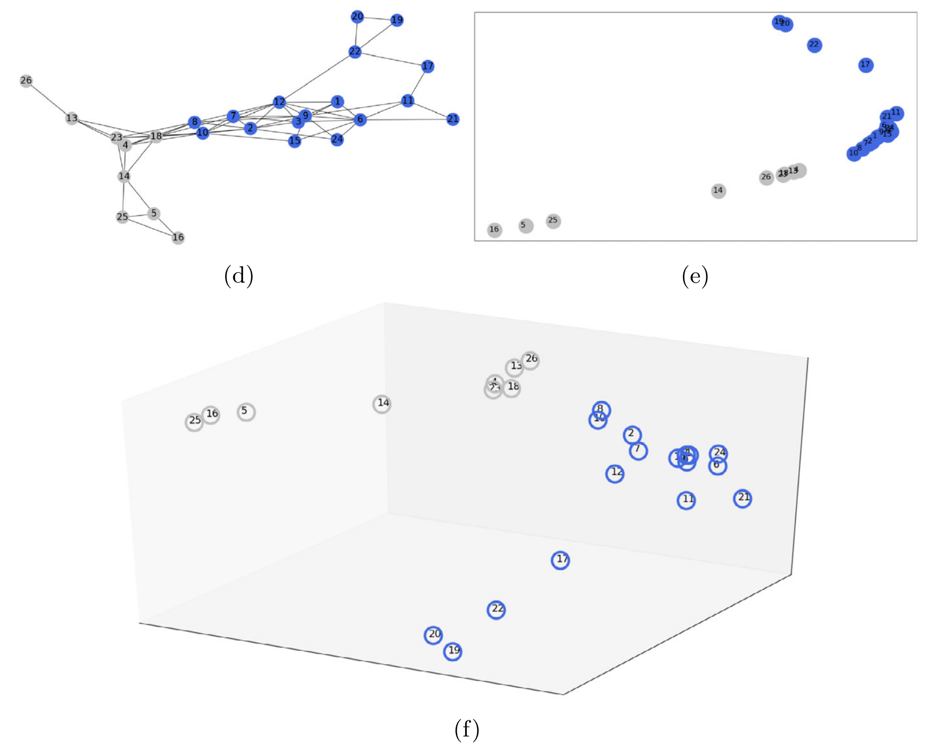


Figure 12 (**a**) Dutch School Friendship Network with 2 communities, (**b**) and (**c**) 2-dimensional and 3-dimensional embedding of the Network using NodeSim Method, respectively

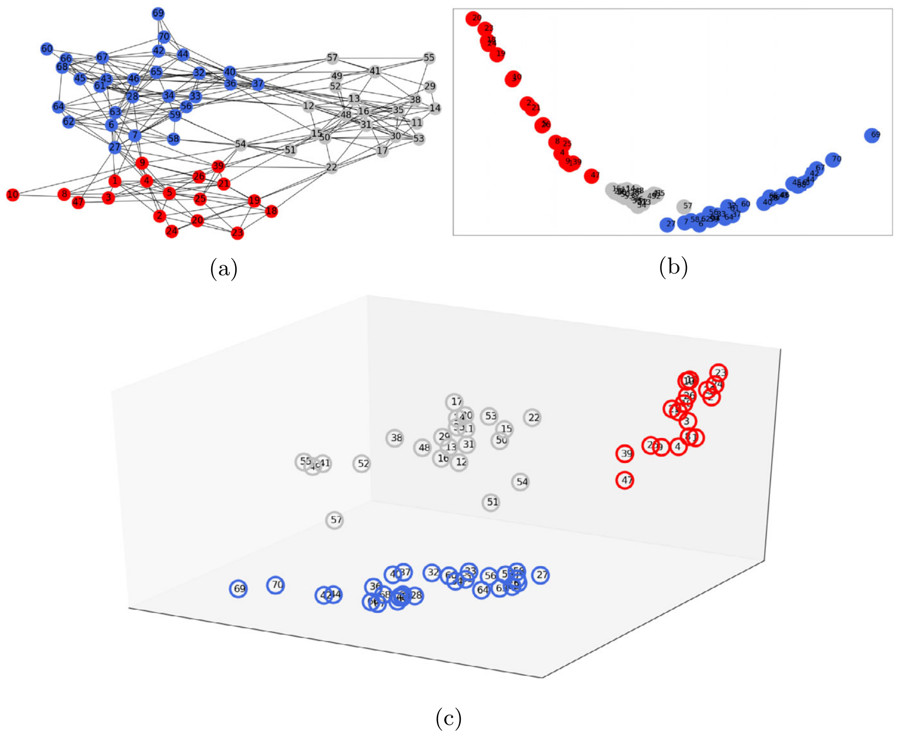


Figure 13 (**a**) Illinois Highschool Friendship Network with three communities, (**b**) and (**c**) 2-dimensional and 3-dimensional embedding of the Network using NodeSim Method, respectively

Appendix B: Embedding visualization

For a better understanding, we further show the 2-dimensional and 3-dimensional NodeSim embedding of the Dutch School Friendship Network [62], and Illinois Highschool Friendship [63] in Fig. 12 and 13, respectively. The Dutch School Friendship network has 26 nodes and 63 edges, and the Highschool network has 70 nodes and 274 edges. In both the figures, the nodes having the same color belong to the same community. In Fig. 13, we can observe that node 57 is well connected with both grey and blue communities, and that is also evident from its embedding. Similar observations can be made for other nodes in both figures.

One another important point to note is that in Fig. 11 for Zachary-Karate network, the 2-dimension embedding is linear; however, this is not the case with Dutch School and Highschool Friendship networks, as the embedding depends on the network connectivity and the proposed method performs well for all kind of networks as we also observed in Sect. 4.3.

Abbreviations
OSNs, Online Social Networks; JC, Jaccard Coefficient; AA, Adamic Adar; RA, Resource Allocation.

Availability of data and materials
The used datasets are publicly available. The implementation code of NodeSim method is available at https://github.com/akratiiet/NodeSim.

Declarations

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

Authors' contributions
All of the authors contributed to designing the proposed model and writing up the results. AS implemented the model and carried out the analysis and experiments. All authors read and approved the final manuscript.

Publisher's Note
Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Received: 16 June 2021 Accepted: 18 March 2022 Published online: 13 April 2022

References

1. Girvan M, Newman ME (2002) Community structure in social and biological networks. Proc Natl Acad Sci 99(12):7821–7826
2. Clauset A, Newman ME, Moore C (2004) Finding community structure in very large networks. Phys Rev E 70(6):066111
3. McPherson M, Smith-Lovin L, Cook JM (2001) Birds of a feather: homophily in social networks. Annu Rev Sociol 27(1):415–444
4. Granovetter M (1983) The strength of weak ties: a network theory revisited. Sociological theory, 201–233
5. Saxena A, Iyengar S (2016) Evolving models for meso-scale structures. In: 2016 8th international conference on communication systems and networks (COMSNETS). IEEE Press, New York, pp 1–8
6. Benevenuto F, Rodrigues T, Cha M, Almeida V (2009) Characterizing user behavior in online social networks. In: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement, pp 49–62
7. Wilson C, Boe B, Sala A, Puttaswamy KP, Zhao BY (2009) User interactions in social networks and their implications. In: Proceedings of the 4th ACM European conference on computer systems, pp 205–218
8. Saxena A, Hsu W, Lee ML, Leong Chieu H, Ng L, Teow LN (2020) Mitigating misinformation in online social network with top-k debunkers and evolving user opinions. In: Companion proceedings of the web conference 2020, pp 363–370
9. Masrour F, Wilson T, Yan H, Tan P-N, Esfahanian A (2020) Bursting the filter bubble: fairness-aware network link prediction. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 841–848
10. Aslay C, Matakos A, Galbrun E, Gionis A (2018) Maximizing the diversity of exposure in a social network. In: 2018 IEEE international conference on data mining (ICDM). IEEE Press, New York, pp 863–868
11. Zhou T, Lü L, Zhang Y-C (2009) Predicting missing links via local information. Eur Phys J B 71(4):623–630
12. Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. J Am Soc Inf Sci Technol 58(7):1019–1031
13. Adamic LA, Adar E (2003) Friends and neighbors on the web. Soc Netw 25(3):211–230
14. Barabási A-L, Albert R (1999) Emergence of scaling in random networks. Science 286(5439):509–512
15. Valverde-Rebaza J, de Andrade Lopes A (2013) Exploiting behaviors of communities of Twitter users for link prediction. Soc Netw Anal Min 3(4):1063–1074
16. Jeon H, Kim T (2017) Community-adaptive link prediction. In: Proceedings of the 2017 international conference on data mining, communications and information technology, pp 1–5
17. Grover A, Leskovec J (2016) node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 855–864
18. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining, pp 701–710
19. Epasto A, Perozzi B (2019) Is a single embedding enough? Learning node representations that capture multiple social contexts. In: The world wide web conference, pp 394–404
20. Valverde-Rebaza J, de Andrade Lopes A (2012) Structural link prediction using community information on Twitter. In: 2012 fourth international conference on computational aspects of social networks (CASoN). IEEE Press, New York, pp 132–137
21. Saxena A, Fletcher G, Pechenizkiy M (2021) Hm-eiict: fairness-aware link prediction in complex networks using community information. Journal of Combinatorial Optimization, 1–18
22. Clauset A, Moore C, Newman ME (2008) Hierarchical structure and the prediction of missing links in networks. Nature 453(7191):98–101
23. Wang C, Satuluri V, Parthasarathy S (2007) Local probabilistic models for link prediction. In: Seventh IEEE international conference on data mining (ICDM 2007). IEEE Press, New York, pp 322–331
24. Scripps J, Tan P-N, Chen F, Esfahanian A-H (2008) A matrix alignment approach for link prediction. In: 2008 19th international conference on pattern recognition. IEEE Press, New York, pp 1–4
25. Menon AK, Elkan C (2011) Link prediction via matrix factorization. In: Joint European conference on machine learning and knowledge discovery in databases. Springer, Berlin, pp 437–452
26. Lu Z, Savas B, Tang W, Dhillon IS (2010) Supervised link prediction using multiple sources. In: 2010 IEEE international conference on data mining. IEEE Press, New York, pp 923–928
27. Benchettara N, Kanawati R, Rouveirol C (2010) A supervised machine learning link prediction approach for academic collaboration recommendation. In: Proceedings of the fourth ACM conference on recommender systems, pp 253–256
28. Kashima H, Kato T, Yamanishi Y, Sugiyama M, Tsuda K (2009) Link propagation: a fast semi-supervised learning algorithm for link prediction. In: Proceedings of the 2009 SIAM international conference on data mining. SIAM, Philadelphia, pp 1100–1111

29. Hu H, Zhu C, Ai H, Zhang L, Zhao J, Zhao Q, Liu H (2017) Lpi-etslp: lncrna–protein interaction prediction using eigenvalue transformation-based semi-supervised link prediction. Mol BioSyst 13(9):1781–1787
30. Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q (2015) Line: large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web, pp 1067–1077
31. Wang D, Cui P, Zhu W (2016) Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 1225–1234
32. Cao S, Lu W, Xu Q (2015) Grarep: learning graph representations with global structural information. In: Proceedings of the 24th ACM international on conference on information and knowledge management, pp 891–900
33. Du L, Lu Z, Wang Y, Song G, Wang Y, Chen W (2018) Galaxy network embedding: a hierarchical community structure preserving approach. In: IJCAI, pp 2079–2085
34. Keikha MM, Rahgozar M, Asadpour M (2018) Community aware random walk for network embedding. Knowl-Based Syst 148:47–54
35. Li Y, Wang Y, Zhang T, Zhang J, Chang Y (2019) Learning network embedding with community structural information. In: IJCAI, pp 2937–2943
36. Ou M, Cui P, Pei J, Zhang Z, Zhu W (2016) Asymmetric transitivity preserving graph embedding. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 1105–1114
37. Ou M, Cui P, Wang F, Wang J, Zhu W (2015) Non-transitive hashing with latent similarity components. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, pp 895–904
38. Lyu T, Zhang Y, Zhang Y (2017) Enhancing the network embedding quality with structural similarity. In: Proceedings of the 2017 ACM on conference on information and knowledge management, pp 147–156
39. Ribeiro LF, Saverese PH, Figueiredo DR (2017) Struc2vec: learning node representations from structural identity. In: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pp 385–394
40. Donnat C, Zitnik M, Hallac D, Leskovec J (2018) Learning structural node embeddings via diffusion wavelets. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp 1320–1329
41. Nikolentzos G, Vazirgiannis M (2019) Learning structural node representations using graph kernels. In: IEEE transactions on knowledge and data engineering
42. Ahmed NK, Rossi RA, Lee JB, Willke TL, Zhou R, Kong X, Eldardiry H (2019) role2vec: role-based network embeddings. In: Proc. DLG KDD, pp 1–7
43. Tu K, Cui P, Wang X, Yu PS, Zhu W (2018) Deep recursive network embedding with regular equivalence. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, pp 2357–2366
44. Newman ME (2006) Modularity and community structure in networks. Proc Natl Acad Sci 103(23):8577–8582
45. Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech Theory Exp 2008(10):10008
46. Onnela J-P, Saramäki J, Hyvönen J, Szabó G, De Menezes MA, Kaski K, Barabási A-L, Kertész J (2007) Analysis of a large-scale weighted network of one-to-one human communication. New J Phys 9(6):179
47. Newman ME (2001) Clustering and preferential attachment in growing networks. Phys Rev E 64(2):025102
48. Lovász L et al (1993) Random walks on graphs: a survey. Combinatorics, Paul erdos is eighty 2(1), 1–46
49. De Winter S, Decuypere T, Mitrović S, Baesens B, De Weerdt J (2018) Combining temporal aspects of dynamic networks with node2vec for a more efficient dynamic link prediction. In: 2018 IEEE/ACM international conference on advances in social networks analysis and mining (ASONAM). IEEE Press, New York, pp 1234–1241
50. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. arXiv preprint. arXiv:1301.3781
51. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: NIPS.
52. Leskovec J, Mcauley JJ (2012) Learning to discover social circles in ego networks. In: Advances in neural information processing systems, pp 539–547
53. Leskovec J, Kleinberg J, Faloutsos C (2007) Graph evolution: densification and shrinking diameters. ACM Trans Knowl Discov Data 1(1):2
54. Klimt B, Yang Y (2004) Introducing the enron corpus. In: CEAS
55. Yang J, Leskovec J (2015) Defining and evaluating network communities based on ground-truth. Knowl Inf Syst 42(1):181–213
56. Gupta Y, Saxena A, Das D, Iyengar S (2016) Modeling memetics using edge diversity. In: Complex networks VII. Springer, Berlin, pp 187–198
57. Saxena A, Iyengar S, Gupta Y (2015) Understanding spreading patterns on social networks based on network topology. In: Proceedings of the 2015 IEEE/ACM international conference on advances in social networks analysis and mining 2015, pp 1616–1617
58. Parés F, Gasulla DG, Vilalta A, Moreno J, Ayguadé E, Labarta J, Cortés U, Suzumura T (2017) Fluid communities: a competitive, scalable and diverse community detection algorithm. In: International conference on complex networks and their applications. Springer, Berlin, pp 229–240
59. Cordasco G, Gargano L (2010) Community detection via semi-synchronous label propagation algorithms. In: 2010 IEEE international workshop on: business applications of social network analysis (BASNA). IEEE Press, New York, pp 1–8
60. Raghavan UN, Albert R, Kumara S (2007) Near linear time algorithm to detect community structures in large-scale networks. Phys Rev E 76(3):036106
61. Zachary WW (1977) An information flow model for conflict and fission in small groups. Journal of anthropological research, 452–473
62. Knecht A, Snijders TA, Baerveldt C, Steglich CE, Raub W (2010) Friendship and delinquency: selection and influence processes in early adolescence. Soc Dev 19(3):494–514
63. Coleman JS et al (1964) Introduction to mathematical sociology. Introduction to mathematical sociology
64. Heidler R, Gamper M, Herz A, Eßer F (2014) Relationship patterns in the 19th century: the friendship network in a German boys' school class from 1880 to 1881 revisited. Soc Netw 37:1–13

65. Müller E, Sánchez PI, Mülle Y, Böhm K (2013) Ranking outlier nodes in subspaces of attributed graphs. In: 2013 IEEE 29th international conference on data engineering workshops (ICDEW). IEEE Press, New York, pp 216–222
66. Metsis V, Androutsopoulos I, Paliouras G (2006) Spam filtering with naive Bayes-which naive Bayes? In: CEAS, Mountain View, CA, vol 17, pp 28–69.
67. Kagan D, Elovichi Y, Fire M (2018) Generic anomalous vertices detection utilizing a link prediction algorithm. Soc Netw Anal Min 8(1):1–13
68. Cao Q, Sirivianos M, Yang X, Pregueiro T (2012) Aiding the detection of fake accounts in large scale social online services. In: 9th USENIX symposium on networked systems design and implementation. NSDI, vol 12, pp 197–210
69. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357