Overlapping Community Detection via Edge-Space Representation

Fengjie Chen[*] **Yikun Zhang**[*]
Department of Statistics,
University of Washington, Seattle
Seattle, WA 98195
`{chenfj3, yikun}@uw.edu`

Abstract

Detecting communities helps researchers in various fields better understand the relationships between members within their domain-specific networks. Traditional community detection algorithms group nodes into clusters or uncover communities based on a narrow neighborhood of each node in the network. In this paper we refine the edge similarity in the Link Clustering[1] in light of edge-space representation so that a broader range of network information can be incorporated. When clustering edges in the network, our proposed *edge2vec* Link Clustering method is able to detect overlapping communities. We evaluate our algorithm on the Amazon and DBLP network data sets with ground-truth community information. Experiments show that our proposed method is comparable to state-of-the-art community detection algorithms.

1 Introduction

Network analysis has become an indispensable approach to understanding the organizations of members in social science[7], interactions of protein-protein in biological networks[20], and investigations of citation and collaborations in scientific disciplines[4]. Community detection is one of the most challenging but actively researched task among network analysis research. A community (also referred to as a module or a cluster) is typically thought of as a group of nodes with more connections among its members than between its members and the remainder of the network.[9] Communities in real-world networks often overlap such that nodes simultaneously belong to several groups.[1, 23] These overlapping structures are conceptually different from hierarchical organizations, where communities are recursively partitioned into several nested clusters.

Even though methods for identifying overlapping as well as hierarchically-nested communities in networks have been intensively researched[1, 17, 2, 23], uncovering meaningful communities in large networks has proven to be a challenging task[24, 6]. There are four prominent overlapping community detection algorithms which outperforms their competitors: Link clustering (LC)[1], Clique Percolation Method (CPM)[17], the Mixed-Membership Stochastic Block Model (MMSB)[2], and Cluster Affiliation Model for Big Networks (BIGCLAM)[23]. From the methodological perspective, link clustering (LC) is an unorthodox approach in the sense that it partitions links (or edges) instead of nodes into clusters. The rest three algorithms, on the contrary, focus on grouping nodes. Clustering over links is beneficial to overlapping community detection, since it reconciles the antagonistic organizing principles of overlapping communities and hierarchy[1].

The originally proposed link clustering approach utilizes a singly-linkage bottom-up hierarchical clustering method with a similarity metric defined as the Jaccard similarity between one-step neighbors of the nodes on the edges that shares a common node. See Section 4.1 for details. The most

[*]Equal contributor. See Section 7 for details.

conspicuous drawback for this similarity metric between edges is that it only takes a narrow scope of networks into consideration. When a micro-view and a macro-view of an edge in the network can provide us with information about the edge from different angles, the original similarity in the link clustering method loses a lot of useful information and can not be adaptive to community detection in miscellaneous networks.

1.1 Present work

Our main goal in this paper is to ameliorate the similarity metric in link clustering so that information in the broader neighborhood of an edge pair can be explored and incorporated in the computation of similarities between edges (or links). We leverage the scalability and flexibility of `node2vec` feature learning mechanism for networks[11] to refine the similarity metric. Given an edge and the `node2vec` feature representations of its endpoints, a binary operation can be applied to compute the edge-space representation of the edge. For unweighted networks, people commonly use the arithmetic mean or Hadamard product to combine node feature representations into their associated edge feature representations. We call this method *edge2vec*, which is different from a previous synonymous method[8]. With the pre-computed edge feature vectors, we naturally define an innovative similarity metric between edges to be the cosine similarity (or normalized inner product). See Section 4.3 for details. Before rigorously carrying out experiments on network data with ground-truth communities, we project the `node2vec` and *edge2vec* feature representations onto the two dimensional space and visualize it with community affiliations via the t-Distributed Stochastic Neighbor Embedding (t-SNE) technique[22]. The visualization indicates that community structures are well-preserved under the latent space representation.

We then apply this similarity metric to link clustering. Note that a single-linkage hierarchical clustering method is not necessary once we obtain *edge2vec* feature representations. Various clustering methods are suitable to clustering edges, which consequently allow for overlapping node communities.

For each detected community an effort is made to interpret it as a “real” community by identifying a common property shared by all the members of the community.[16] Such evaluation procedures require extensive manual effort, which are non-comprehensive and limited to small networks. In order to evaluate existing community detection methods with our modified link clustering, we retrieve two network data sets with naturally arisen ground-truth communities from `http://snap.stanford.edu`. Specifically, we conduct our experiments on the Amazon product co-purchasing network and DBLP collaboration network and compare our proposed method with two overlapping community detection algorithms, CPM and BIGCLAM. We omit the comparison with MMSB due to the fact that MMSB is not scalable to such large networks and also empirically proven to be inferior than BIGCLAM. The evaluations are based on *Average F1 Score* and *Omega Index* described in Yang and Leskovec [23, 24].

2 Related Work

Community detection has been extensively studied in the last decades, and a number of algorithms have been developed to detect non-overlapping communities. These community detection algorithms essentially fall into two categories, heuristic methods (e.g. GN[9]) and optimization-based approaches (e.g. Girvan-Newman (GN) fast[12]). In addition, Louvain method[5] is a greedy algorithm that partitions a network into communities by maximizing a quantity called Modularity, which was introduced by Girvan and Newman [9]. On the other hand, Andersen et al. [3] computes the approximate personalized Page-Rank vectors and cuts a graph based on conductance.

These graph partition algorithms, however, cannot be directly adaptive to discover overlapping communities[1]. In order to discover overlapping structures of networks, many algorithms have been proposed, which can be roughly divided into two categories: node-based and link-based algorithms.

The node-based algorithms cluster nodes of network directly, utilizing the structural information of nodes. For instance, researchers propose a fuzzy community detection algorithm that calculates the possibility of each node belonging to every community.[14, 13, 2] One downside of these algorithms is that the number of communities has to be determined in advance. There are some overlapping community detection algorithms that are able to automatically determine the number of communities. BIGCLAM[23] holds out 20% of node pairs and chooses the number of communities by maximizing the likelihood on the hold-out set. Meanwhile, it formulates community detection as a variant of nonnegative matrix factorization (NMF) and learns latent factors which represent community affiliations of nodes.[23] Another model called Clique Percolation Method (CPM)[17] starts from defining a k-clique (or complete graph with k nodes) and identifies sets of nodes that can be reached through a sequence of adjacent k-cliques as communities. Nevertheless, searching for those sets of nodes is considered as an NP-hard problem, which can be hardly generalized to large-scale network data.

The most well-known link-based method was proposed by Ahn et al. [1] and applied by Q.Ye et al. [19] to massive networks. This innovative method clusters the links in a network into communities, and then derive the corresponding node communities from link clusters based on the incident relationship between edges and nodes. It creates the brand new avenue to study complex community structure of networks. However, as we mentioned in the previous section 1, the main drawback of the link clustering is that its proposed similarity metric only considers a narrow scope of neighborhoods, which could be further improved by *edge2vec* edge-space representations.

Our *edge2vec* approach builds on top of an algorithmic framework for learning continuous feature representations for nodes in networks. Recently, deep neural network based representation learning has been actively studied in the fields of Natural Language Processing. In particular, the Skip-gram model[15] aims to learn continuous feature representations for words by optimizing a neighborhood preserving likelihood objective. Inspired by the Skip-gram model, recent research established an analogy for networks by generating ordered sequences of nodes that serve as the neighborhoods of nodes.[11, 18, 21] For instance, DeepWalk[18] and LINE[21] learn a d-dimensional feature representations by simulating uniform random walks or explore immediate and two-hop neighbors of nodes so as to sample the ordered sequence of nodes. The `node2vec` avenue outweighs other feature learning methods by offering some flexibility in sampling of nodes from a network. Besides inheriting the framework of the Skip-gram model, `node2vec` introduces two parameters p and q to bias the random walks and hence interpolate between Breath-first Sampling and Depth-first Sampling.

3 Data Description

The upcoming comparisons between our proposed *edge2vec* method and the existing well-known algorithms are based on two network data sets with ground-truth community labels, Amazon product co-purchasing network and DBLP collaboration network. They are both undirected and unweighted networks with ground-truth communities. (Note that our proposed *edge2vec* community detection method can be generalized to tackling either directed or weighted network data.) We conduct our experiments only on these two network data sets due to the limit of project duration and considerable time needed to carry out thorough experiments.

In the Amazon network, each node represents a product with an unique product ID and each edge connects commonly co-purchased products. Each product (*i.e.*, node) belongs to one or more hierarchically nested product categories, which defines ground-truth communities. Members of the same community share a common function or a role. Ground-truth communities in the Amazon network can be overlapping or hierarchically nested.[24]

We also use the collaboration network of DBLP (a computer science bibliography website), where nodes represents authors/actors and edges connect nodes that have co-authored a paper. The publication venues, such as journals or conferences, are considered as ground-truth communities in DBLP and serve as proxies for highly overlapping scientific communities around which the collaboration network then organizes.[24] Table 1 summarizes some basic statistics of these two network data sets.

Table 1: Basic Statistics of Network Data Sets

Network	**Nodes**	**Edges**	**C**	**ACS**	**CM**
Amazon	334863	925872	75149	30.22	6.78
DBLP	317080	1049865	13477	53.40	2.27

*C: number of communities, ACS: average community size, CM: community memberships per node.

The histograms of node degrees for the two networks are shown in Figure 1. It illustrates that two networks share similar patterns in their node degree distributions with heavy right tails. Note that there are no isolated nodes (with 0 degrees) in both network data sets.

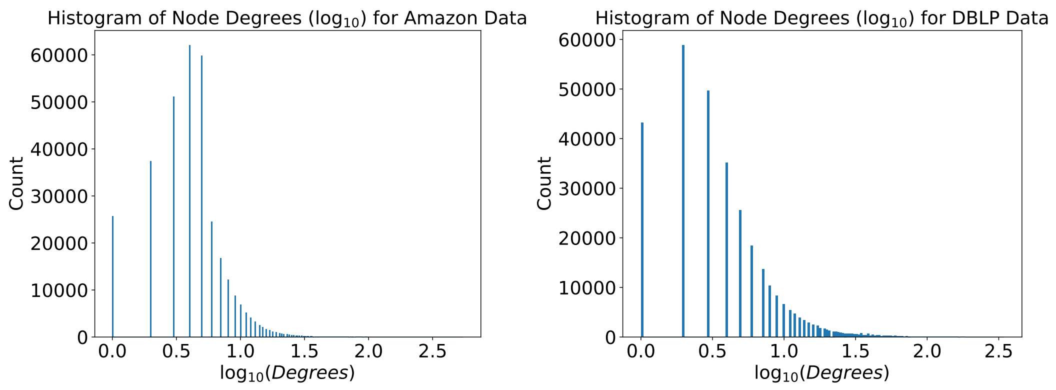


Figure 1: Degree Distributions for Amazon and DBLP networks.

4 Method

4.1 Link Clustering

In order to group the links in a network into clusters, we need to define a similarity metric as well as a clustering method. The original similarity metric proposed by Ahn et al. [1] limits to link pairs that share a node, which are expected to be more similar than disconnected pairs. For an undirected, unweighted network, we denote the set of node i and its neighbors as $n_+(i)$. The similarity S between links (or edges) e_{ik} and e_{jk} is defined to be[1]

$$S(e_{ik}, e_{jk}) = \frac{|n_+(i) \cap n_+(j)|}{|n_+(i) \cup n_+(j)|},$$

when e_{ik} and e_{jk} share a common node k. The similarity becomes 0 when two links have no common nodes.

Based on this similarity metric, a single-linkage hierarchical clustering is applied to build a dendrogram. Cutting the dendrogram at some clustering threshold, it yields link communities. The choice of the threshold is critical to the outcome of any clustering method. In this case, we choose the threshold with maximum partition density.

partition density[1] For a network with M links and N nodes, $P = \{P_1, P_2, ..., P_C\}$ is a partition of links into C subsets. The subset P_c has $m_c = |P_c|$ links and $n_c = |\cup_{e_{ij} \in P_c} \{i, j\}|$ induced nodes. Then the link density D_c of community c is

$$D_c = \frac{m_c - (n_c - 1)}{n_c(n_c - 1)/2 - (n_c - 1)}.$$

This is the number of links in P_c normalized by the minimum and maximum numbers of links possible between n_c connected nodes. (We assume that $D_c = 0$ if $n_c = 2$) The partition density, D, is the average of D_c, weighted by the fraction of present links:

$$D = \frac{1}{M} \sum_c m_c D_c.$$

This strategy of selecting the threshold for hierarchical clustering methods will remain unchanged after we refine the similarity metric. Now we are supposed to discuss how learning feature representations for nodes engages in the formulation of similarity between edges.

4.2 `Node2vec` Feature Learning

As an algorithmic frameworks for learning continuous feature representations for nodes in networks, the `node2vec` model generates a mapping of nodes to a low-dimensional space of features that maximizes the likelihood of preserving network neighborhoods of nodes.[11] Essentially, `node2vec` extends the Skip-gram architecture to networks.[15]

4.2.1 Technical Background

Let $G = (V, E)$ be a given network and $f : V \to \mathbb{R}^d$ be the mapping function from nodes to feature representations. The `node2vec` model seeks to maximize the log-probability of observing a network neighborhood $N_S(u)$ for a node u conditioned on its feature representation, given by f:

$$\max_f \sum_{v \in V} \log Pr(N_S(u)|f(u)).$$

The conditional independence assumption is made to simplify the optimization problem and for any $n_i \in N_S(u)$, the conditional likelihood of every source-neighborhood node pairs is modeled as a softmax unit in the following way:

$$Pr(n_i|f(u)) = \frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u)}.$$

In order to sample neighborhoods of a source node as a form of local search, Grover and Leskovec [11] develop a flexible biased random walk procedure that can smoothly interpolate between Breadth-first Sampling (BFS) and Depth-first Sampling (DFS). Given a source node u as an initial state c_0, a random walk of fixed length l will be generated by the following distribution:

$$P(c_i = x|c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v, x) \in E, \\ 0 & \text{otherwise,} \end{cases}$$

where π_{vx} is the unnormalized transition probability between nodes v and x, and Z is the normalizing constant. The unnormalized transition probability π_{vx} incorporates two parameters p and q, which control how fast the walk explores and leaves the neighborhood of starting node u. Consider a random walk that just traversed edge (t, v) and now resides at node v. Then the unnormalized transition probability to next node x is defined to be $\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$, where

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0, \\ 1 & \text{if } d_{tx} = 1, \\ \frac{1}{q} & \text{if } d_{tx} = 2, \end{cases}$$

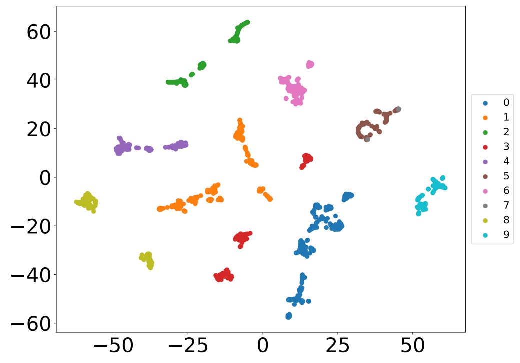
and $d_{tx} \in \{0, 1, 2\}$ denotes the shortest path distance between nodes t and x. By setting the transition probability in this way, the random walks are 2^{nd} order Markovian.

4.2.2 `node2vec` on Amazon and DBLP Data sets

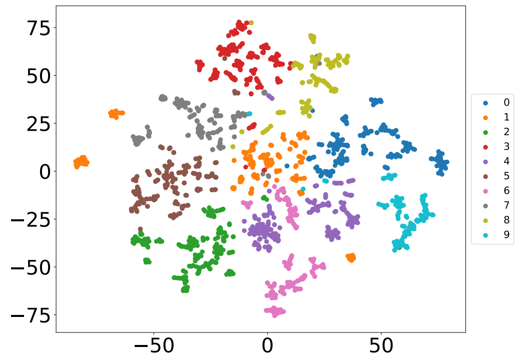
As a task-independent feature representation learning framework, `node2vec` can be utilized to tackle any downstream network analysis task. This setting, on one hand, equips `node2vec` with great flexibility in dealing with real-world network related problems. On the other hand, the flexibility of `node2vec` comes at risk, in the sense that it may not be tailored to a particular network analysis problem and degrade the predictability of any model built on `node2vec` features.

In order to empirically justify the correctness of `node2vec` representations on our community detection task, we apply the `node2vec` algorithm on Amazon and DBLP datasets. After learning the 128-dimension embedding vector for each node in two graphs, we visualize the nodes of 10 randomly selected gound-truth communities. Since visualizations in any space with dimension higher than three are almost impossible, a standard high-dimensional data visualization technique called "t-SNE" (t-Distributed Stochastic Neighbor Embedding) is applied. This technique is not only easier to optimize when compared to Stochastic Neighbor Embedding but can produce significantly better visualizations by reducing the tendency to crowd points together in the center of the two or three-dimensional map.[22] Moreover, the main topological structure of `node2vec` representations is well-preserved when they are projected to two-dimensional spaces. Figure 2 illuminates the t-SNE visualizations of `node2vec` representations on Amazon and DBLP data sets.

In Figure 2, when represented by `node2vec` vectors and projected to two-dimensional spaces via t-SNE, nodes within each community tend to be closed to each other and nodes from different communities are relatively far away from each other. This phenomenon is particularly conspicuous on the sampled community of the Amazon network and a majority of nodes in the sampled DBLP network can be distinguished from communities to communities. Therefore, it is convincing that the *node2vec* representation framework is suitable for community detection on our data sets.



(a) Amazon



(b) DBLP

Figure 2: Visualization of *node2vec* representations on Amazon and DBLP Datasets via t-SNE

4.3 *Edge2vec* Feature Representation

After learning `node2vec` representations for all the nodes in a graph, it is natural to define the similarity for pair of nodes by the usual cosine similarity,

$$S(i,j) = \frac{f(i)^T f(j)}{||f(i)||_2 \cdot ||f(j)||_2},$$

where $|| \cdot ||_2$ is the L^2 norm and $f(i), f(j) \in \mathbb{R}^d$ are *node2vec* representations of nodes i and j, respectively. Based on the cosine similarity metric, any state-of-the-art clustering methods can be applied to group nodes into communities. Hence the `node2vec` feature learning framework provides us a possible venue to carry out community detection on any given dataset.

However, community detection relied on `node2vec` representations, though reasonable and powerful in some cases, embraces a salient drawback, that is, it can hardly deal with cases when overlapping communities are present. As mentioned earlier, we are supposed to inherit the mechanism of link clustering. In order to bridge the connection between `node2vec` representations and link clustering, two components are indispensable, an edge representation learning approach and similarity metric for links (or edges).

4.3.1 Mathematical Background

We seek to preserve the powerful property of `node2vec` feature representation when it comes to community detection. Thus, our *edge2vec* feature representation is built on top of `node2vec` algorithmic framework. Given two nodes i and j and their node2vec feature vector $f(i), f(j)$, the *edge2vec* representation of e_{ij} is defined to be

$$vec(e_{ij}) = f(i) \circ f(j),$$

where the binary operator $\circ$ can be the arithmetic mean, Hadamard product, or weighted-L1/L2 operations.[11] As for our experiments in this paper, we insist on the arithmetic mean approach to calculate *edge2vec* representations. Depending on the real-world scenarios, other binary operations might be more appropriate.

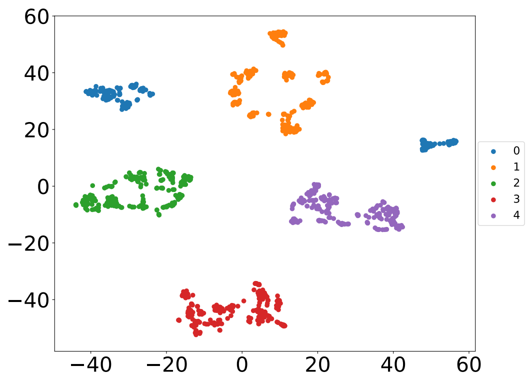
After the definition of *edge2vec* representations is clear, the similarity metric between pair of adjacent edges e_{ik}, e_{jk} can be computed as *node2vec* representations, *i.e.*, the cosine similarity (or normalized inner product)

$$S(e_{ik}, e_{jk}) = \frac{vec(e_{ik})^T vec(e_{jk})}{||vec(e_{ik})||_2 \cdot ||vec(e_{jk})||_2},$$

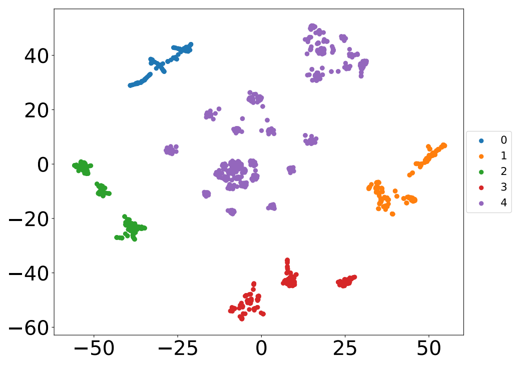
where $vec(e_{ik}) \in \mathbb{R}^d$ is the *edge2vec* feature vector of e_{ik}. If two edges share no common nodes, the similarity between them is again 0.

4.3.2 *Edge2vec* on Amazon and DBLP datasets

Similarly, we conduct some visualizations of *edge2vec* representation based on ground-truth community labels so as to verify the validity of applying *edge2vec* to community detection. As implemented in the `node2vec` part, we utilize t-SNE to project the learned *edge2vec* vectors of Amazon and DBLP data sets to two-dimensional spaces. See Figure 3 for details.



(a) Amazon



(b) DBLP

Figure 3: Visualization of *edge2vec* representations on Amazon and DBLP Datasets via t-SNE

Figure 3 again serves as a more convincing evidence that *edge2vec* representations have the capability to distinguish communities from each other. In both Amazon and DBLP datasets, the *edge2vec* vectors within community are gathered together while edges from different communities tend to stay away from each other when projected to lower dimensional spaces.

4.4 *Edge2vec* Link Clustering Algorithm

For each network data, we first run a *node2vec* algorithm to obtain feature vectors of each node in the network. Then the feature vector of each edge (i, j) is computed as an arithmetic mean of feature vectors of i and j. In order to improve the performance of link clustering, we replace its original similarity metric with cosine similarity between edge feature representations. Except for the edge similarity, we conscientiously follow the single-linkage hierarchical clustering method and cut the dendrogram based on maximum partition density. After node communities are identified by our modified link clustering method, we use two evaluation metrics defined below to quantify its accuracy and compare with other state-of-the-art overlapping community detection algorithms.

4.5 Evaluation Metrics

The availability of network data with ground-truth communities enable us to *quantitatively* evaluate the performance of community detection algorithms. For evaluation we use metrics that quantify the level of correspondence between the detected and the ground-truth communities.[23] Given network $G = (V, E)$, we denote C^* as a set of ground-truth communities and $\hat{C}$ as a set of detected communities, where each $C_i \in C^*$ and each $\hat{C}_i \in \hat{C}$ is a set of its member nodes. To measure how similar $\hat{C}$ is to C^*, we consider:

Average F1 score[23] We define F1 score to be the average of the F1-score of the best-matching ground-truth community to each detected community, and the F1-score of the best-matching detected community to each ground-truth community:

$$\frac{1}{2}\left[\frac{1}{|C^*|}\sum_{C_i \in C^*} F1(C_i, \hat{C}_{g(i)}) + \frac{1}{|\hat{C}|}\sum_{\hat{C}_i \in \hat{C}} F1(C_{g'(i)}, \hat{C}_i)\right],$$

where the best matching g and g' is defined as follows:

$$g(i) = \arg\max_j F1(C_i, \hat{C}_j), \quad g'(i) = \arg\max_j F1(C_j, \hat{C}_i)$$

and $F1(C_i, \hat{C}_j)$ is the harmonic mean of Precision and Recall.

Omega Index[10] is the accuracy on estimating the number of communities that each pair of nodes shares:

$$\frac{1}{|V|^2} \sum_{u,v \in V} \mathbf{1}\{|C_{uv}| = |\hat{C}_{uv}|\}$$

where C_{uv} is the set of ground-truth communities that nodes u and v share and $\hat{C}_{uv}$ is the set of detected communities that they share.

For these two metrics higher values are equivalent to more "accurately" detected communities. Maximum value of 1 is attained when the detected communities perfectly corresponds to the ground-truth communities.

5 Results

We proceed by evaluating the performance of our modified link clustering method and comparing it to some prominent baseline community detection algorithms on the Amazon and DBLP network data set. These baseline methods are the original Link Clustering (LC)[1], Clique Percolation Method (CPM)[17], and Cluster Affiliation Model for Big Networks (BIGCLAM)[23].

We use the Python implementation for the original link clustering provided by the author on his GitHub: `https://github.com/bagrow/linkcomm`. For CPM and BIGCLAM, we run the Python code in the Stanford Network Analysis Platform (SNAP)[2]. For CPM, we use the clique size $k = 3$. For BIGCLAM, the number of communities is automatically determined by maximizing the likelihood on the hold out set and the maximal number of communities to try is 10000. Our `node2vec` implementation also relies on this SNAP platform and the open source code on GitHub: `https://github.com/aditya-grover/node2vec`. To be consistent with the experiments in the original paper[11], we set the parameters for `node2vec` as follows.

- Dimension of feature vectors: 128
- The number of simulated random walks per node: 20
- The length of each random walk: 80
- The context window size (inheriting from the Skip-gram model): 10
- The number of iterations in the optimization step: 20
- The return parameter p: 0.4
- The in-out parameter q: 1.0

Unless mentioned explicitly, the parameter choices will be default values for all aforementioned methods in the subsequent experiments.

5.1 Experimental Setup

Given an unlabeled undirected network G (with hold out known ground-truth communities C^*) we aim to uncover communities $\hat{C}$ such that $\hat{C}$ closely match the ground-truth communities C^*.

Based on our experiments, BIGCLAM, original LC, and our proposed method can process our large-scale network data sets (contradictory to the claim in Yang and Leskovec [23]). However, CPM cannot scale to networks of such size. Therefore, when comparing our modified LC to BIGCLAM and the original LC, we conduct the experiments on the whole Amazon and DBLP networks. To allow for comparison between our method and CPM, we apply the following simulation strategy described in Yang and Leskovec [23], where the goal is to obtain a large set of relatively small subnetworks with overlapping community structure. For each graph G, we pick a random node u that belongs to at least two communities. We then take the subnetwork to be the induced subgraph of G consisting of all the nodes that share at least one ground-truth community membership with u. Since on average 95% of all ground-truth communities overlap[23], this simulation procedure does not bias towards overlapping communities and we can easily obtain 100 different subnetworks for each network data.

[2]http://snap.stanford.edu/

5.2 Evaluation Details

In our experiments, we believe that informative and useful communities should have at least 3 members, so we discard the detected communities with size less than 3 when computing *Average F1 score* and *Omega Index*. In addition, computing *Average F1 score* needs $O(m_1 m_2)$ time, and computing *Omega Index* needs $O(n^2)$ time, where $m_1 = |C^*|$ is the number of ground-truth communities, $m_2 = |\hat{C}|$ is the number of detected communities, and $n = |V|$ is the total number of nodes in a network. For our two network data sets, $m_1 \geq 10000, m_2 \approx 100000, n \geq 300000$, so computing evaluation metrics is substantially time-consuming.

To tackle this issue, a Monte Carlo method is introduced. Suppose K detected communities are sampled uniformly from m_2 detected communities. For each sample $\hat{C}_k$, $F1(C_{g'(k)}, \hat{C}_k)$ is computed and denoted as $x_k^{(2)}$. Our estimate of *Average F1 score* becomes

$$\frac{1}{2}\left[x^{(1)} + \frac{1}{K}\sum_{k=1}^{K} x_k^{(2)}\right],$$

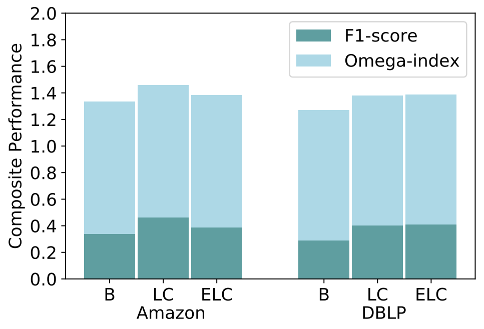
where $x^{(1)} = \frac{1}{|C^*|}\sum_{C_i \in C^*} F1(C_i, \hat{C}_{g(i)})$ is the true average F1-score from the best-matching of ground-truth communities. If we denote the true average F1-score from the best-matching of detected communities by $x^{(2)}$, then by Chebyshev's inequality,

$$P\left[\frac{1}{2}(x^{(1)} + \frac{1}{K}\sum x_k^{(2)}) - \frac{1}{2}(x^{(1)} + x^{(2)}) > \epsilon\right] = P(|\bar{x}^{(2)} - x^{(2)}| > 2\epsilon) \leq \frac{Var(x_k^{(2)})}{K \cdot 4\epsilon^2} \leq \frac{1}{4K\epsilon^2}.$$

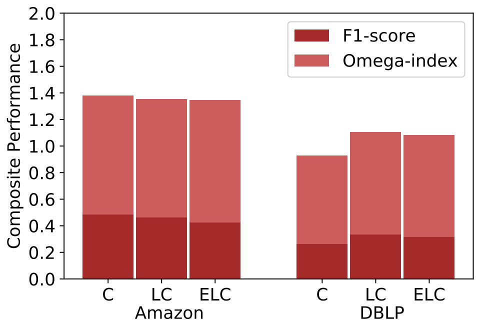
If we want the probability to be less than 0.05 and pick $\epsilon = 0.05$, we need $K \geq 2000$. That is, with probability greater than 95%, our Monte Carlo estimate lies within 0.05-range of true value if we only sample 2000 true communities for calculating the best-matching in the true community pool. The same error estimate inequality applies to the calculations of the best-matching for true communities to detected communities. Using the similar argument, we sample 50000 pairs of nodes to compute *Omega Index*, which ensures that with probability greater than 95% our estimate lies within 0.02-range from the true *Omega Index*.

5.3 Experimental Results

For the BIGCLAM, original LC, and our *edge2vec* LC, we measure the average F1-score and Omega Index on the whole Amazon and DBLP network data sets. For the CPM, original LC, and our *edge2vec* LC, we compute the average value of these two metrics over the 100 subnetworks using the strategy described in Section 5.1. Then we display the composite performance by summing up the average F1-score and Omega Index for each method in Figure 4.



(a) Metrics on the whole network



(b) Average metrics on the sampled subnetworks

Figure 4: Composite Performances of Methods on Amazon and DBLP Network. B: BIGCLAM; LC: Link Clustering; ELC: *edge2vec* Link Clustering; C: Clique percolation method (CPM).

When applied on the whole network data set, our proposed *edge2vec* link clustering method outperforms the effective BIGCLAM algorithm in terms of both the average F1-score and Omega Index.

Moreover, the composite performance of our *edge2vec* link clustering is around 1.38 on the DBLP data set, which slightly outweighs the original link clustering method. As for simulated subnetworks, our *edge2vec* link clustering has the average composite score 1.09, which is almost 8.5% higher than CPM (0.92) on the DBLP data set. The reason why our *edge2vec* link clustering method embraces less decent performance on the Amazon data set is that the Amazon network is sparser on edges but have more labeled communities. When fewer edges are present, the information retained in *edge2vec* representations tends to be less sufficient. However, given the fact that we have not conduct thorough experiments to figure out the optimal binary operation from `node2vec` to *edge2vec*, the results here are promising and have great research potential.

In order to present the capability of our proposed *edge2vec* link clustering method to uncover overlapping communities, we visualize a partial Amazon network with detected communities from our method in Figure 5.

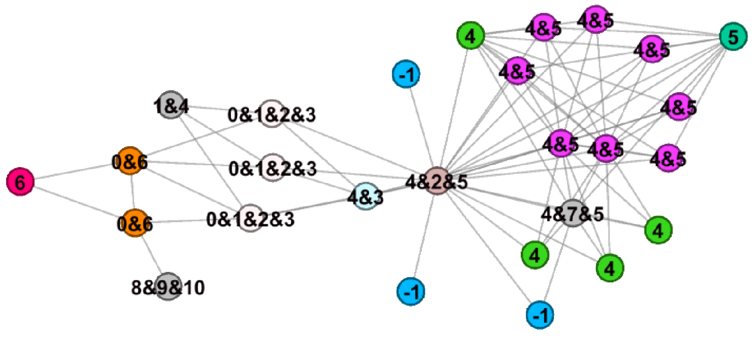


Figure 5: Partial Detected Communities via our *edge2vec* LC Method on the Amazon Network

6 Conclusion and Future Work

In this paper we proposed a framework that incorporates task-independent feature representation learning for nodes in a network into community detection. When network structures are manipulated to *edge2vec* latent space representations, the community structure in the original network is well-preserved. By utilizing cosine similarity to embed *edge2vec* representations into the definition of edge similarity in the conventional Link Clustering method, our proposed *edge2vec* Link Clustering method not only has the capability to detect overlapping community structures but is also scalable to large networks with millions of nodes and edges. Experiments on the Amazon and DBLP network data sets indicate that our proposed method is comparable to the existing popular community detection algorithms and can even outperform them when the network data is more compact and has comparatively fewer number of communities.

Our work sheds light on the possibility of combining unsupervised feature learning on graphs with overlapping community detection. One potential future direction is to investigate the optimal operation for transforming from the node-space representation to corresponding edge-space representation. For instance, the semantics of edges can be taken into account during feature representation learning.[8] It enables us to further improve the performance of our *edge2vec* link clustering method under miscellaneous scenarios. Meanwhile, when two-dimensional *edge2vec* visualizations with ground-truth communities yields promising results, it is possible to project the *edge2vec* feature vectors to lower dimensional spaces before computing edge similarity. This pre-processing procedure is worth being scrutinized in order to release the magic of edge-space feature learning.

7 Individual Contributions

Fengjie Chen: Problem formulation, coming up with the algorithm idea, implementing the original Link Clustering and our *edge2vec* Link Clustering, coding up evaluation metrics, tabulating final results, and carrying out the detected community visualization for our method, responsible for the final report writeups, etc.

Yikun Zhang: Problem formulation, coming up with the algorithm idea, implementing the `node2vec`, CPM, and BIGCLAM methods, coding up the subnetwork simulation strategy, plotting graphs for data descriptions and t-SNE visualizations of `node2vec` and *edge2vec* representations, responsible for the report and poster writeups, etc.

References

[1] Y.-Y. Ahn, J. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 2010. ISSN 0028-0836.

[2] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*, 9:1981–2014, 2008.

[3] R. Andersen, F. Chung, and K. Lang. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486, Oct 2006. doi: 10.1109/FOCS.2006.44.

[4] L. Backstrom, D. Huttenlocher, J. Kleinberg, and X. Lan. Group formation in large social networks: Membership, growth, and evolution. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 44–54, New York, NY, USA, 2006. ACM.

[5] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008. URL `http://stacks.iop.org/1742-5468/2008/i=10/a=P10008`.

[6] T. Chakraborty, A. Dalmia, A. Mukherjee, and N. Ganguly. Metrics for community analysis: A survey. *ACM Comput. Surv.*, 50(4):54:1–54:37, Aug. 2017.

[7] S. L. Feld. The focused organization of social ties. *American Journal of Sociology*, 86(5): 1015–1035, 1981. URL `http://www.jstor.org/stable/2778746`.

[8] Z. Gao, G. Fu, C. Ouyang, S. Tsutsui, X. Liu, and Y. Ding. edge2vec: Learning node representation using edge semantics. *CoRR*, abs/1809.02269, 2018. URL `http://arxiv.org/abs/1809.02269`.

[9] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, June 2002.

[10] S. Gregory. Fuzzy overlapping communities in networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2011(02):P02017, feb 2011. doi: 10.1088/1742-5468/2011/02/p02017.

[11] A. Grover and J. Leskovec. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 855–864, New York, NY, USA, 2016. ACM.

[12] R. Guimera and L. A. N. Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, Feb. 2005. ISSN 0028-0836.

[13] T. S. K. Nowicki. Estimation and prediction for stochastic blockstructures. *Journal of the American Statistical Association 96 (455) 1077–1087*, 2001.

[14] M. Magdon-Ismail and J. T. Purnell. Ssde-cluster: Fast overlapping clustering of networks using sampled spectral distance embedding and gmms. In *SocialCom/PASSAT*, pages 756–759. IEEE Computer Society, 2011.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[16] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review*, E 69(026113), 2004.

[17] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, June 2005. ISSN 0028-0836.

[18] B. Perozzi, R. Al-Rfou, and S. Skiena. DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710. ACM, 2014.

[19] Q.Ye, B.Wu, Z.X.Zhao, and B.Wang. Detecting link communities in massive networks. *ASONAM,71–78*, 2011.

[20] P. Radivojac, W. Clark, T. Oron, A. M Schnoes, T. Wittkop, A. Sokolov, K. Graim, C. Funk, K. Verspoor, A. Ben-Hur, G. Pandey, J. Yunes, A. S Talwalkar, S. Repo, M. L Souza, D. Piovesan, R. Casadio, Z. Wang, J. Cheng, and I. Friedberg. A large-scale evaluation of computational protein function prediction. *Nature Methods*, 10:221–227, 01 2013. doi: 10.1038/nmeth.2340.

[21] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. LINE: Large-scale Information Network Embedding. In *WWW*, 2015.

[22] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL `http://www.jmlr.org/papers/v9/vandermaaten08a.html`.

[23] J. Yang and J. Leskovec. Overlapping community detection at scale: A nonnegative matrix factorization approach. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 587–596, New York, NY, USA, 2013. ACM.

[24] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. *Knowl. Inf. Syst.*, 42(1):181–213, Jan. 2015. ISSN 0219-1377.