A Unified Framework for Community Detection and Network Representation Learning

Cunchao Tu*, Xiangkai Zeng*, Hao Wang, Zhengyan Zhang, Zhiyuan Liu, Maosong Sun, Bo Zhang, and Leyu Lin

Abstract—Network representation learning (NRL) aims to learn low-dimensional vectors for vertices in a network. Most existing NRL methods focus on learning representations from local context of vertices (such as their neighbors). Nevertheless, vertices in many complex networks also exhibit significant global patterns widely known as communities. It's intuitive that vertices in the same community tend to connect densely and share common attributes. These patterns are expected to improve NRL and benefit relevant evaluation tasks, such as link prediction and vertex classification. Inspired by the analogy between network representation learning and text modeling, we propose a unified NRL framework by introducing community information of vertices, named as Community-enhanced Network Representation Learning (CNRL). CNRL simultaneously detects community distribution of each vertex and learns embeddings of both vertices and communities. Moreover, the proposed community enhancement mechanism can be applied to various existing NRL models. In experiments, we evaluate our model on vertex classification, link prediction, and community detection using several real-world datasets. The results demonstrate that CNRL significantly and consistently outperforms other state-of-the-art methods while verifying our assumptions on the correlations between vertices and communities.

Index Terms—Network Representation Learning, Community Detection, Link Prediction, Social Networks.

1 Introduction

Network is an important way to organize relational information such as social relations, citations and other interactions among multiple objects. With the rapid development of large-scale online social networks such as Facebook, Twitter, and Weibo, network data is constantly growing and gives rise to many challenges on dealing with these large-scale real-world network data.

How to represent network data is critical when applying machine learning algorithms to network analysis tasks, such as vertex classification, personalized recommendation, anomaly detection and link prediction [1], [2], [3]. Conventional graph-based representation regards each vertex as a discrete symbol, which does not consider the semantic relations between vertices and usually suffers from the sparsity issue.

In recent years, network representation learning (NRL) has been widely adopted to network analysis, which aims to build low-dimensional vectors for vertices according to their structural roles in networks. With the learnt real-valued low-dimensional representations, NRL enables us to measure the semantic relations between vertices efficiently and also alleviates the sparsity issue in conventional graph-based representation.

Most NRL methods learn vertex representations according to their *local context* information. For example, DeepWalk [4] performs random walks over the network topology and learns vertex representations by maximizing the likelihood of predicting their local contextual vertices in walk sequences; LINE [5] models first and second order proximities of vertex pairs, and learns vertex representations by maximizing the likelihood of predicting their neighbor vertices. Both *contextual vertices* in DeepWalk and *neighbor vertices* in LINE are local context. Although some extensions of DeepWalk and LINE (e.g., node2vec [6] and GraRep [7]) can explore a broader range of contextual information, the considered structural information is still limited and can only reflect local patterns.

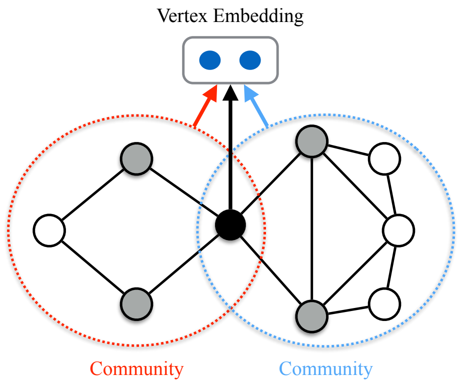


Fig. 1. The idea of Community-enhanced NRL: each vertex may belongs to multiple communities and the embedding of a specific vertex is determined by both its neighbor vertices and communities.

As illustrated in Fig. 1, in a typical complex network, vertices usually group into multiple communities with each community densely connected inside [8]. Vertices in a community usually share certain common attributes. For example, Facebook users

- *Cunchao Tu, Hao Wang, Zhengyan Zhang, Zhiyuan Liu (corresponding author), and Maosong Sun are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China.*
Email: tucunchao@gmail.com, {hwang12, zhangzhengyan14}@mails.tsinghua.edu.cn, {liuzy, sms}@tsinghua.edu.cn
- *Xiangkai Zeng is with the School of Computer Science, Beihang University, Beijing 100084, China.*
Email: kevinwirehack@gmail.com
- *Bo Zhang and Leyu Lin are with the Search Product Center, WeChat Search Application Department, Tencent, Beijing 100080, China.*
Email: {nevinzhang, goshawklin}@tencent.com
- ** indicates equal contribution.*

Manuscript received April 19, 2005; revised August 26, 2015.

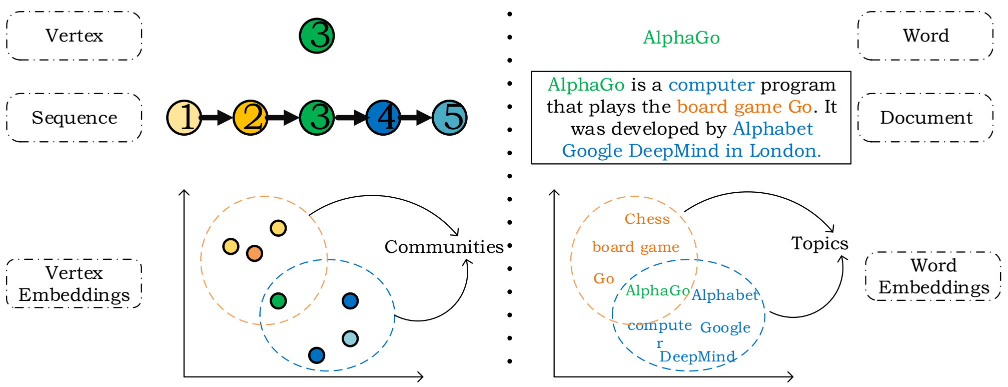


Fig. 2. The analogy between network representation learning and text modeling. Vertices, sequences, and communities in a network correspond to words, documents(or sentences), and topics in natural language respectively.

with the same education-based attributes ("School name" or "Major") tend to form communities [9]. Hence, the community structure is an important *global pattern* of vertices, which is expected to benefit NRL as well as network analysis tasks. Inspired by this, we propose a unified framework for community detection and network representation learning, named as Community-enhanced NRL (CNRL).

It is worth pointing out that CNRL is highly motivated by the connection between text modeling and network modeling as well as DeepWalk [4]. In DeepWalk, the authors find that the vertex frequency in random walk sequences follows the power-law distribution, which is same as word frequency distribution in the text corpus. Therefore, by regarding vertices as words and vertex sequences as text, they directly re-purpose language modeling methods (i.e., skip-gram [10]) to model random walk sequences and learn vertex representations. Furthermore, as illustrated in Fig. 2, we extend the analogy between networks and natural languages by assuming that there exists an intermediate state between random walk sequences and vertices, i.e., communities, which corresponds to the topics between text and words. This assumption is quite intuitive and has been verified through further experiments and analysis. Based on this assumption, we propose CNRL to detect communities and learn network representations by employing topic modeling methods. Although community information has been explored by some NRL models (e.g., MNMF [11] and ComE [12]), we employ an entirely different approach to solve this problem by leveraging the analogy between topics and communities. This analogy is explicit and makes CNRL easy to be integrated into random-walk based NRL models.

The basic idea of CNRL is demonstrated in Fig. 3. We consider each vertex is grouped into multiple communities, and these communities are overlapping. Different from conventional NRL methods, where the vertex embedding is learnt from local context vertices, CNRL will learn the vertex embedding from both local context and global community information.

In CNRL, it is crucial to determine which community each vertex in a sequence belongs to. As the analogy between words in text and vertices in walk sequences has been verified by [4], we assume that there are correlations between word preference on topics and vertex preference on communities as well. Following the idea of topic models, each vertex in a specific sequence is assigned with a specific community, according to the community distribution of the vertex and that of the sequence. Afterwards, each vertex and its assigned community are applied to predict its context vertices in the walk sequence. Therefore, representations of both vertices and communities are learnt by maximizing the prediction likelihood. Note that community distributions of vertices are also updated iteratively in the representation learning process.

The community representations learnt in CNRL will serve to enhance vertex representations in network analysis tasks, such as vertex classification and link prediction. More importantly, community representations are expected to benefit those long-tail vertices with less local context information.

We implement CNRL on two typical random-walk based NRL models, DeepWalk [4] and node2vec [6]. Since both components of CNRL can be accelerated by existing parallel algorithms, it takes only linear time to train CNRL in real-world scenarios. For comparison, we conduct experiments on several real-world network datasets using the tasks of vertex classification, link prediction, and community detection. Experimental results show that CNRL can significantly improve the performance of all the tasks, and the superiority is consistent with respect to various datasets and training ratios. It demonstrates the effectiveness of considering global community information for network representation learning.

To summarize, our major contributions are as follows:

(1) We exploit the analogy between topics in text and communities in networks, and introduce community information, a critical network analysis factor, into NRL to learn discriminative network representations. The proposed CNRL model is able to detect community structures and learn representations of vertices jointly.

(2) We propose two efficient community assignment strategies of CNRL, statistics-based one and embedding-based one. The two implementations are from different views and work equally well. Moreover, CNRL can be trained in linear due to the existing accelerated algorithms of both components.

(3) For evaluation, we conduct two typical network analysis tasks, vertex classification and link prediction on a variety of real-world networks at different scales. The experimental results show that CNRL achieves significant and consistent improvements than state-of-the-art methods.

(4) More specifically, our proposed CNRL can detect overlapping communities on multiple scales. To demonstrate the flexibility of CNRL, we visualize the detected communities of different scales and make comparisons to typical community detection method.

2 Related Work

Network representation learning (NRL, i.e., network embedding) aims to encode the structure and other additional information of vertices into real-valued low-dimensional representation space. Afterwards, the learnt representations will be treated as features and fed into further network analysis applications, including vertex classification, link prediction, community detection, and so on. In the following parts of this section, we will introduce related work on these highly relevant tasks and NRL respectively.

2.1 Vertex Classification

As one of the most important network analysis tasks, vertex classification aims to classify each vertex into one or several categories according to its features, especially using features of network topology. It corresponds to many real-world applications, such as user profiling [13], [14], anomaly detection [15], [16], [17], [18], and so on.

The performance of vertex classification highly depends on the quality of vertex features. Traditional methods usually employ hand-crafted features that relevant to the category systems (e.g., age [19], gender [20], profession [21], personality [22] et. al). However, these methods usually need many human efforts and lack of capability to other real-world scenarios.

To address these challenges, researchers propose NRL to construct effective features automatically [4], [5]. Moreover, the low-dimensional representations in NRL models also overcome the sparsity issue and are more efficient for computation. It's also worth noting that many semi-supervised NRL models are proposed to improve the particular vertex classification task, including MMDW [23], DDRW [24], Planetoid [25], and GCN [26]. These methods can incorporate labeling information during the learning process, and obtain discriminative representations of vertices.

2.2 Link Prediction

Link prediction is a fundamental task in network analysis, which aims to predict if there exists an edge between two vertices. Generally, it measures the similarity of two vertices and gives potentially connected vertices a higher score than irrelevant vertices.

Traditional methods use hand-crafted similarity measurements such as common neighbors [27], Salton index [28], Jaccard index and resource allocation index [29] for link prediction. Besides these topology-based methods, link propagation methods [30], [31] are based on graph regularization, which is a common approach in semi-supervised learning. Matrix factorization methods [32] predict links by adapting matrix completion technic to the adjacency matrix. Further, people employ tensor to develop temporal model [33] for link prediction on dynamic networks. Other methods improve the effect of link prediction by taking information like community affinity [34] and vertex attribute [35] into account.

2.3 Community Detection

According to the definition in [36], a community in a network is a group of vertices, within which vertices are densely connected but between which they are linked sparsely. In real-world social networks, vertices in the same community usually share common properties or play similar roles [37].

Detecting communities from networks is a critical research filed in social science. In terms of community detection, traditional methods focus on partitioning the vertices into different groups, i.e., detecting non-overlapping communities. Existing non-overlapping community detection works mainly include clustering-based methods [38], modularity based methods [8], [37], [39], spectral algorithms [40], stochastic block models [41], [42] and so on. The major drawback of these traditional methods is that they cannot detect overlapping communities, which may not accord with real-world scenarios. To address this problem, CPM [43] is proposed to generate overlapping communities by merging overlapping k-cliques. Ahn [44] proposes link clustering for overlapping community detection by employing non-overlapping community detection methods to partition the links instead of vertices and then assigning a single vertex to corresponding groups of its links.

In recent years, community affiliation based algorithms show their effectiveness on overlapping community detection [45], [46], [47], [48]. Community affiliation based algorithms predefine the number of communities and learn a vertex-community strength vector for each vertex and assign communities to vertices according to the vector. For example, Yang [48] proposes Non-negative Matrix Factorization (NMF) method, which approximates adjacency matrix $A \in R^{|V|\times|V|}$ by FF^T where matrix $F \in R^{|V|\times k}$ is vertex-community affinity matrix. Then the algorithm learns non-negative vertex embeddings and converts each dimension of the embeddings into a community. These community affiliation based algorithms try to approximate the adjacency matrix in value and design different objective functions for it. Our model follows the idea to represent the community relationship with a non-negative vector, but we don't explicitly set hard community membership for the vertices.

2.4 Network Representation Learning

Representation learning has shown its effectiveness in computer vision [49] and natural language processing [10]. Representation learning is also becoming an important technique for network analysis in recent years. Current methods [4], [5], [50], [51] embed each vertex into a real-valued vector space based on modeling local information and take the representations as features in further evaluation tasks.

More specifically, social dimension [50] computes the top-d eigenvectors of adjacency matrix and takes them as d-dimensional representations. By first generating random walks from a network, DeepWalk [4] employs word2vec [10], a widely-used word representation learning algorithm, to learn vertex embeddings from random walk sequences. Comparing with DeepWalk, node2vec [6] designs an effective random walk strategy for sequence generation based on BFS and DFS. LINE [5] characterizes the probability of first-order and second-order proximities between two vertices. GraRep [7] models k-order proximity between two vertices through matrix factorization. Struc2vec [52] constructs a multilayer context graph to encode structural similarities between vertices and generate structural context for them. Some works employ deep neural networks to learn vertex representations, such as deep auto-encoders [53], [54], convolutional neural networks [26], [55] and deep generation models [56].

Vertices in real-world networks usually accompany with heterogeneous information, such as text contents and labels. There are a variety of works that attempt to incorporate this information into

NRL. TADW [51] and CANE [57] extend existing NRL models to take advantage of the accompanying text information. By utilizing labeling information of vertices, MMDW [23] employs max-margin principle to learn discriminative network representations. Chang [58] design a deep embedding architecture for capturing complex heterogeneous data in a network. TransNet [59] employs translation mechanism to model the interactions among vertices and labels on edges. GCN [26] and GraphSAGE [55] takes additional features of vertices as inputs and generate different levels of vertex representations layer-by-layer.

Most existing NRL methods only consider the local neighbors of vertices, and ignore the global patterns, such as community structure. Although random walk based methods (e.g., DeepWalk and node2vec) can explore a broad range of connected vertices and traverse in closely connected subgraphs, the exploration of contextual information is restricted by the window size, according to the proof in [51], [60]. Moreover, community information is not utilized explicitly in these models. [11] proposes modularized nonnegative matrix factorization (MNMF) model to detect non-overlapping communities for improving vertex representations. However, there are two disadvantages of MNMF. First, it can only detect non-overlapping communities (i.e., each vertex only belongs to a specific community), which is usually not in conformity with real-world networks. Besides, MNMF is a matrix factorization based model, and its optimization complexity is $\mathcal{O}(n^2m + n^2k)$ (n is the vertex number, m is the dimension of vertex representation, and k is the community number), which makes it hard to handle large-scale networks. [12] also proposes ComE to learn vertex embedding and detect communities simultaneously. Specifically, each community in ComE is represented as a multivariate Gaussian distribution to model how its member vertices are distributed. With well-designed iterative optimization algorithms, ComE can be trained efficiently with the complexity that is linear to the graph size. However, ComE only attempts to find the optimal multivariate Gaussian distribution of each community to fit the learnt vertex embeddings, without considering the explicit network structure of communities. In this work, we exploit the analogy between topics in text and communities in networks and propose a novel NRL model, CNRL, which can be easily and efficiently incorporated into existing NRL models. To the best of our knowledge, our model is the first attempt to learn community-enhanced network representations by utilizing the analogy between topics and communities.

3 Community-enhanced NRL

We start with discussing the necessary notations and formalizations of NRL.

3.1 Formalizations

We denote a network as $G = (V, E)$, where V is the set of vertices and $E \subseteq (V \times V)$ is the set of edges, with $(v_i, v_j) \in E$ indicating there is an edge between v_i and v_j. For each vertex v, NRL aims to learn a low-dimensional vector denoted as $\mathbf{v} \in \mathbb{R}^d$. Here d represents the dimension of representation space.

The vertices in G can be grouped into K communities $C = \{c_1, \dots, c_K\}$. The communities are usually overlapping. That is, one vertex may be the member of multiple communities in different degrees. Hence, we record the membership degree of a vertex v to a community c as the probability $\Pr(c|v)$, and the role of the vertex in c as the probability $\Pr(v|c)$. In this work, we will also learn representations of each community c, denoted as $\mathbf{c}$.

In the following part, we first give a brief introduction to DeepWalk. Afterwards, we implement the idea of CNRL by extending DeepWalk to Community-enhanced DeepWalk. As the only difference between node2vec and DeepWalk is the generation methods of vertex sequences, which will not affect the extension of CNRL, we omit its implementation details.

3.2 DeepWalk

DeepWalk [4] performs random walks over the given network G firstly, and forms a set of walk sequences $S = \{s_1, \dots, s_N\}$, where each sequence can be denoted as $s = \{v_1, \dots, v_{|s|}\}$.

DeepWalk treats each walk sequence s as a word sequence by regarding vertices as words. By introducing Skip-Gram [61], a widely-used word representation learning algorithm, DeepWalk is able to learn vertex representations from the sequence set S.

More specifically, given a vertex sequence $s = \{v_1, \dots, v_{|s|}\}$, each vertex v_i has $\{v_{i-t}, \dots, v_{i+t}\} \setminus \{v_i\}$ as its local context vertices. Following Skip-Gram, DeepWalk learns vertex representations by maximizing the average log probability of predicting context vertices:

$$\mathcal{L}(s) = \frac{1}{|s|} \sum_{i=1}^{|s|} \sum_{i-t \le j \le i+t, j \ne i} \log \Pr(v_j|v_i), \tag{1}$$

where v_j is the context vertex of the vertex v_i, and the probability $\Pr(v_j|v_i)$ is defined by softmax function:

$$\Pr(v_j|v_i) = \frac{\exp(\mathbf{v}'_j \cdot \mathbf{v}_i)}{\sum_{v \in V} \exp(\mathbf{v}' \cdot \mathbf{v}_i)}. \tag{2}$$

Note that, as in Skip-Gram, each vertex v in DeepWalk also has two representation vectors, i.e., $\mathbf{v}_i$ when it is the center vertex and $\mathbf{v}'$ when it is the context vertex.

3.3 Community-enhanced DeepWalk

With random walk sequences, DeepWalk aims to maximize the local conditional probability of two vertices within a context window. That means, the co-occurrence of vertices in a sequence only relies on the affinity between vertices, while ignoring their global patterns. A critical global pattern of social networks is homophily, i.e., “birds of a feather flock together” [62]. That is, those similar vertices sharing the same “feather” may group into communities. It’s worth pointing out that such global pattern has not been considered by DeepWalk.

Communities provide rich contextual information of vertices. In order to take community information into consideration to provide a richer context for NRL, we make two assumptions on the correlations among vertices, walk sequences and communities.

Assumption 1: Each vertex in a network may belong to multiple communities with different preferences, i.e., $\Pr(c|v)$, and each vertex sequence also owns its community distribution $\Pr(c|s)$.

With the above assumption, we make another assumption about the particular community of a vertex in a sequence.

Assumption 2: A vertex in a specific walk sequence belongs to a distinct community, and the community is determined by the sequence’s distribution over communities $\Pr(c|s)$ and the community’s distribution over vertices $\Pr(v|c)$.

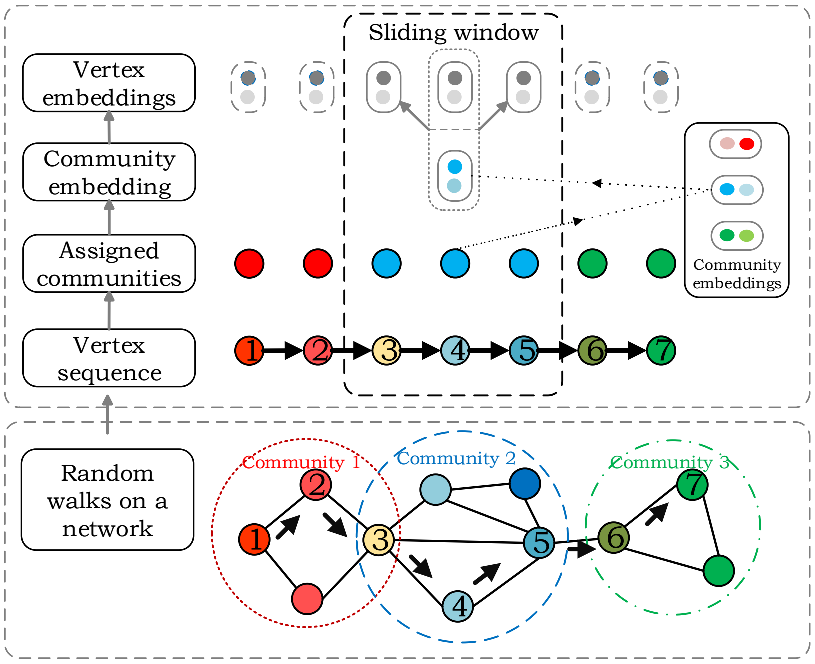


Fig. 3. An illustration of CNRL.

With the above assumptions and generated random walk sequences, we conduct the following two steps iteratively to detect community structures and learn representations of vertices and communities: (1) **Community Assignment.** We assign a discrete community label for each vertex in a particular walk sequence, according to both local context and global community distribution. (2) **Representation Learning.** Given a vertex and its community label, we learn optimized representations to maximize the log probability of predicting context vertices.

The two steps are demonstrated in Fig. 3. As shown in Fig. 3, we aim to learn an embedding for each vertex and each community. Besides, we also want to learn the community distribution of each vertex. We introduce the two steps in detail as follows.

3.3.1 Community Assignment

For a vertex v in a walk sequence s, we calculate the conditional probability of a community c as follows:

$$\Pr(c|v,s) = \frac{\Pr(c,v,s)}{\Pr(v,s)} \propto \Pr(c,v,s). \tag{3}$$

According to our assumptions, the joint distribution of (c,v,s) can be formalized as

$$\Pr(c,v,s) = \Pr(s)\Pr(c|s)\Pr(v|c), \tag{4}$$

where $\Pr(v|c)$ indicates the role of v in the community c, and $\Pr(c|s)$ indicates the local affinity of the sequence s with the community c. From Eq. (3) and Eq. (4), we have

$$\Pr(c|v,s) \propto \Pr(v|c)\Pr(c|s). \tag{5}$$

In this work, we propose the following two strategies to implement $\Pr(c|v,s)$:

Statistic-based assignment. Following the Gibbs Sampling method of Latent Dirichlet Allocation (LDA) [63], we can estimate the conditional distributions of $\Pr(v|c)$ and $\Pr(c|s)$ as follows:

$$\Pr(v|c) = \frac{N(v,c)+\beta}{\sum_{\tilde{v}\in V} N(\tilde{v},c) + |V|\beta}, \tag{6}$$

$$\Pr(c|s) = \frac{N(c,s)+\alpha}{\sum_{\tilde{c}\in C} N(\tilde{c},s) + |K|\alpha}. \tag{7}$$

Here $N(v,c)$ indicates how many times the vertex v is assigned to the community c, and $N(c,s)$ indicates how many vertices in the sequence s are assigned to the community c. Both $N(v,c)$ and $N(c,s)$ will be updated dynamically as community assignments change. Moreover, α and β are smoothing factors following [63].

Embedding-based assignment. As CNRL will obtain the embeddings of vertices and communities, we can measure the conditional probabilities from an embedding view instead of global statistics. Hence, we formalize $\Pr(c|s)$ as follows:

$$\Pr(c|s) = \frac{\exp(\mathbf{c}\cdot\mathbf{s})}{\sum_{\tilde{c}\in C}\exp(\tilde{\mathbf{c}}\cdot\mathbf{s})}, \tag{8}$$

where $\mathbf{c}$ is the community vector learnt by CNRL, and $\mathbf{s}$ is the semantic vector of the sequence s, which is the average of the embeddings of all vertices in s.

In fact, we can also calculate $\Pr(v|c)$ in the similar way:

$$\Pr(v|c) = \frac{\exp(\mathbf{v}\cdot\mathbf{c})}{\sum_{\tilde{v}\in V}\exp(\tilde{\mathbf{v}}\cdot\mathbf{c})}. \tag{9}$$

However, the usage of Eq. (9) will badly degrade the performance. We suppose the reason is that vertex embedding is not exclusively learnt for measuring community membership. Hence, Eq. (9) could not be as discriminative as compared to the statistic-based Eq. (6). Therefore, in embedding-based assignment, we only calculate $\Pr(c|s)$ using embeddings and still use statistic-based $\Pr(v|c)$.

With estimated $\Pr(v|c)$ and $\Pr(c|s)$, we assign a discrete community label c for each vertex v in sequence s according to Eq. (5).

3.3.2 Representation Learning of Vertices and Communities

Given a certain vertex sequence $s = \{v_1, \ldots, v_{|s|}\}$, for each vertex v_i and its assigned community c_i, we will learn representations of both vertices and communities by maximizing the log probability of predicting context vertices using both v_i and c_i, which is formalized as follows:

$$\mathcal{L}(s) = \frac{1}{|s|} \sum_{i=1}^{|s|} \sum_{i-t \le j \le i+t, j \ne i} \log \Pr(v_j|v_i) + \log \Pr(v_j|c_i), \tag{10}$$

where $\Pr(v_j|v_i)$ is identical to Eq. (2), and $\Pr(v_j|c_i)$ is calculated similar to $\Pr(v_j|v_i)$ using a softmax function:

$$\Pr(v_j|c_i) = \frac{\exp(\mathbf{v}'_j \cdot \mathbf{c}_i)}{\sum_{\tilde{v} \in V} \exp(\tilde{\mathbf{v}}' \cdot \mathbf{c}_i)}. \tag{11}$$

In practice, we employ negative sampling [4] to approximate this probability.

3.3.3 Enhanced Vertex Representation

After the above-mentioned representation learning process, we will obtain representations of both vertices and communities, as well as community distribution of vertex, i.e. $\Pr(c|v)$. We can apply these results to build enhanced representation for each vertex, denoted as $\hat{\mathbf{v}}$. The enhanced representations encode both local and global context of vertices, which are expected to promote discriminability of network representation.

Specifically, $\hat{\mathbf{v}}$ consists of two parts, the original vertex representation $\mathbf{v}$ and its community representation $\mathbf{v}_c$, where

$$\mathbf{v}_c = \sum_{\tilde{c} \in C} \Pr(\tilde{c}|v)\tilde{\mathbf{c}}. \tag{12}$$

Afterwards, we concatenate these two parts and obtain $\hat{\mathbf{v}} = \mathbf{v} \oplus \mathbf{v}_c$.

The detailed pseudocode is shown in Algorithm 1.

Algorithm 1 Training Process of CNRL

```
Require: graph $G = (V, E)$, community size $K$, window size $t$
Ensure: vertex embedding $\mathbf{v}$, context embedding $\mathbf{v}'$, community
    distribution $\Pr(v|c)$, community embedding $\mathbf{c}$
 1: $S \leftarrow$ SamplePath($G$)
 2: Initialize $\mathbf{v}$ and $\mathbf{v}'$ by Skip-Gram with $S$
 3: Assign a community for each vertex in $\mathcal{S}$ randomly
 4: for $iter = 1 : L$ do
 5:     for each vertex $v_i$ in each sequence $s \in S$ do
 6:         Calculate Eqs. (6) and (7) w/o current assignment
 7:         Assign a community $c_i$ for $v_i$ by Eq. (5)
 8:     end for
 9: end for
10: while not convergent do
11:     for each vertex $v_i$ in each sequence $s \in S$ do
12:         Calculate Eq. (5) with Eq. (7) or Eq. (8)
13:         Assign a community $c_i$ for $v_i$ by Eq. (5)
14:         for each $j \in [i - t : i + t]$ do
15:             SGD on $\mathbf{v}$, $\mathbf{v}'$ and $\mathbf{c}$ by Eqs. (10) and (11)
16:         end for
17:     end for
18: end while
```

3.4 Complexity Analysis

In CNRL, the training process consists of two parts, i.e., Gibbs Sampling of LDA and representation learning. Note that, the embedding-based assignment also requires a pre-trained LDA to calculate the statistic-based $\Pr(v|c)$. The complexity of Gibbs Sampling based LDA is $\mathcal{O}(LKnw\gamma)$, where L is the loop size, K is the community size, n is the number of vertices, w is the random walk sequences of each vertex and γ is the sequence length. For representation learning, the complexity is $\mathcal{O}(n \log n)$ for S-DW and E-DW or $\mathcal{O}(n \log n + na^2)$ for S-n2v and E-n2v. Here, a is the average degree of vertices. In total, the complexity of CNRL is $\mathcal{O}(n(LKw\gamma + \log n + a^2))$. In practice, both components of CNRL can be accelerated by parallel algorithms, such as PLDA [64], PLDA+ [65] and Skip-Gram [61].

4 Experiments

In experiments, we adopt the tasks of vertex classification and link prediction to evaluate the performance of vertex embeddings. Besides, we also investigate the effectiveness of our model for community detection.

4.1 Datasets

TABLE 1
Statistics of the real-world networks.

Datasets	Cora	Citeseer	Wiki	BlogCatalog
# Vertices	2,708	3,312	2,405	10,312
# Edges	5,429	4,732	15,985	333,983
# Labels	7	6	19	47
Avg.Degree	4.01	2.86	6.65	32.39

We conduct experiments on four widely adopted network datasets, including Cora, Citeseer, Wiki and BlogCatalog.

Cora. Cora[1] is a research paper set constructed by [66]. It contains $2,708$ machine learning papers which are categorized into 7 classes. The citation relationships between them form a typical social network.

Citeseer. Citeseer is another research paper set constructed by [66]. It contains $3,312$ publications and $4,732$ connections between them. These papers are from 6 classes.

Wiki. Wiki [67] contains $2,405$ web pages from 19 categories and $15,985$ links between them. It's much denser than Cora and Citeseer.

BlogCatalog. BlogCatalog [50] is a social network among blog authors. The bloggers also provide their interested topics, which are regarded as labels.

More detailed information about these datasets is listed in Table 1. These networks belong to different categories and own certain characteristics, such as sparsity and amount of labels.

Besides, we also conduct experiments on a toy network named Zachary's Karate network [68] to visualize detected communities by our model. Karate network is a social network of friendships between a karate club members. It contains 34 vertices and 78 edges.

4.2 Baseline Methods

We employ six state-of-the-art NRL models as baselines, including **DeepWalk**, **LINE**, **node2vec**, **SDNE**, **MNMF**, and **ComE**.

DeepWalk: DeepWalk is an efficient network representation learning model proposed by [4]. It performs random walks on

1. https://people.cs.umass.edu/ mccallum/data.html

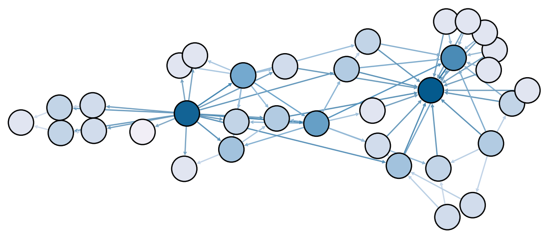


Fig. 4. Karate network. The color of each vertex indicates the its degree. The stronger the background color is, the larger the degree of this vertex is.

networks to generate vertex sequences. With these sequences, DeepWalk adopts a widely used word representation learning model, Skip Gram [61], to learn vertex representations.

LINE: LINE [5] is another network representation learning model that can handle large-scale social networks with millions of vertices. In LINE, the first-order proximity and second-order proximity are proposed to represent the joint and conditional probabilities of vertex pairs respectively. In experiments, we employ both first-order (denoted as LINE-1st) and second-order (denoted as LINE-2nd) LINE as baselines.

node2vec: Node2vec [6] is an extension of DeepWalk. With the utilization of BFS and DFS strategies, node2vec designs a biased random walk strategy to generate vertex sequences and achieves significant improvements than DeepWalk.

SDNE: SDNE [53] is the first attempt to employ deep neural networks (i.e., deep auto-encoder) to construct vertex embeddings.

MNMF: MNMF [11] jointly detects non-overlapping communities and learns network representations through matrix factorization. The learnt representations are expected to preserve the community information.

ComE: ComE [12] learns network representation and detects overlapping communities jointly. Specifically, it employs a multivariate Gaussian distribution to represent each community.

Besides, we also employ four popular link prediction methods as baselines, which are mainly based on local topological properties [69]:

Common Neighbors (CN). For vertex v_i and v_j, CN [27] simply counts the common neighbors of v_i and v_j as similarity score:

$$sim(v]_i, v_j) = |N_i^+ \cap N_j^+|. \tag{13}$$

Salton Index. For vertex v_i and v_j, Salton index [28] further considers the degree of v_i and v_j. The similarity score can be formulated as:

$$sim(v_i, v_j) = (|N_i^+ \cap N_j^+|)/(\sqrt{|N_i^+| \times |N_j^+|}). \tag{14}$$

Jaccard Index. For vertex v_i and v_j, Jaccard index is defined as:

$$sim(v_i, v_j) = (|N_i^+ \cap N_j^+|)/(|N_i^+ \cup N_j^+|). \tag{15}$$

Resource Allocation Index (RA). RA index [29] is the sum of resources received by v_j:

$$sim(v_i, v_j) = \sum_{v_k \in N_i^+} \frac{1}{|N_k^+|}. \tag{16}$$

For community detection, we select four most prominent methods as baselines:

Sequential Clique Percolation (SCP) [70] is a faster version of Clique Percolation [43], which detects communities by searching for adjacent cliques.

Link Clustering (LC) [44] aims to find link communities rather than nodes.

MDL [42] employs the minimum description length principle (MDL) to perform model selection over various stochastic block models.

BigCLAM [46] is a typical nonnegative matrix factorization based model which can detect densely overlapping and hierarchically nested communities in massive networks.

4.3 Parameter Settings and Evaluation Metrics

As mentioned in the previous section, we implement CNRL on DeepWalk and node2vec. Taking DeepWalk for example, we denote the statistic-based and embedding-based community-enhanced DeepWalk as S-DW and E-DW respectively. Similarly, the corresponding implementations of node2vec are denoted as S-n2v and E-n2v.

For a fair comparison, we apply the same representation dimension as 128 in all methods. In LINE, as suggested in [5], we set the number of negative samples to 5 and the learning rate to 0.025. We set the number of side samples to 1 billion for BlogCatalog and 10 million for other datasets. For random walk sequences, we set the walk length to 40, the window size to 5 and the number of sequences started from each vertex to 10. Besides, we employ grid-search to obtain the best performing hyper-parameters in MNMF.

Note that, the representation vectors in CNRL consist of two parts, including the original vertex vectors and the corresponding community vectors. For a fair comparison, we set the dimension of both vectors to 64 and finally obtain a 128-dimensional vector for each vertex. Besides, the smoothing factor α is set to 2 and β is set to 0.5.

For vertex classification, as each vertex in Cora, Citeseer and Wiki has only one label, we employ L2-regularized logistic regression (L2R-LR), the default setting of Liblinear [71] to build classifiers. For multi-label classification in BlogCatalog, we train one-vs-rest logistic regression, and employ *micro-F1* for evaluation.

For link prediction, we employ a standard evaluation metric **AUC** [72]. Given the similarity of all vertex pairs, AUC is the probability that a random unobserved link has higher similarity than a random nonexistent link. Assume that we draw n independent comparisons, the AUC value is $(n_1 + 0.5n_2)/n$, where n_1 is the times that unobserved link has a higher score and n_2 is the times that they have an equal score.

For community detection, we employ **modified modularity** [73] to evaluate the quality of detected overlapping communities.

4.4 Vertex Classification

In Table 2, Table 3, Table 4, and Table 5, we show the vertex classification results under different training ratios. For each training ratio, we randomly select vertices as training set and the rest as the test set. Note that, we employ smaller training ratios on BlogCatalog to accelerate the training speed of multi-label classifiers and evaluate the performance of CNRL under sparse scenes. From these tables, we have the following observations:

TABLE 2
Vertex classification accuracies (%) on Cora

%Training ratio	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	70.77	73.35	74.53	74.94	75.62	76.07	76.08	76.33	77.27
LINE	70.61	74.79	76.93	77.99	78.66	79.22	79.53	79.35	79.67
node2vec ($p = 2, q = 0.5$)	73.29	76.03	77.52	78.08	78.40	78.51	78.72	79.06	79.15
SDNE	70.97	75.08	76.90	77.82	78.26	79.11	79.37	79.46	79.37
MNMF	75.08	77.85	79.05	79.53	79.82	80.21	79.98	80.11	79.41
ComE	**76.72**	79.25	80.73	80.97	81.53	82.10	82.19	82.42	82.65
S-DW	72.78	75.93	77.47	77.98	78.69	79.14	79.15	78.99	78.23
E-DW	73.35	76.56	77.11	78.63	79.18	79.86	79.96	79.94	80.26
S-n2v ($p = 4, q = 1$)	75.86	**79.92**	**81.21**	**82.13**	**82.81**	**83.06**	**82.95**	**83.78**	**83.65**
E-n2v ($p = 4, q = 1$)	76.30	79.40	80.62	81.19	81.46	81.82	81.67	82.16	82.80

TABLE 3
Vertex classification accuracies (%) on Citeseer

%Training ratio	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	47.92	51.54	52.92	54.14	54.21	54.58	55.07	56.09	55.33
LINE	44.27	47.57	50.10	51.15	51.93	52.74	53.46	53.98	54.01
node2vec ($p = 2, q = 0.25$)	49.47	53.27	54.22	55.51	55.87	56.34	56.95	57.61	57.56
SDNE	47.35	51.10	52.45	53.20	53.70	54.20	54.79	55.26	54.46
MNMF	51.62	53.80	55.47	56.94	56.81	57.04	57.05	57.00	57.22
ComE	**54.71**	**57.70**	**58.84**	**59.67**	59.93	60.30	61.12	61.62	61.11
S-DW	49.40	52.58	54.83	55.92	56.63	56.99	57.46	58.48	58.14
E-DW	49.48	52.52	54.41	55.29	56.25	56.21	57.14	57.53	57.41
S-n2v ($p = 4, q = 1$)	53.12	56.68	58.20	59.48	**60.31**	**60.76**	**61.21**	**61.90**	**62.63**
E-n2v ($p = 4, q = 1$)	51.84	54.33	55.85	56.47	57.19	57.11	57.85	58.67	58.51

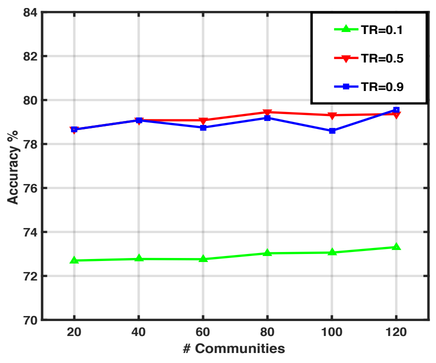
TABLE 4
Vertex classification accuracies (%) on Wiki

%Training ratio	10%	20%	30%	40%	50%	60%	70%	80%	90%
DeepWalk	58.54	62.12	63.56	65.22	65.90	66.53	67.22	67.50	67.56
LINE	57.53	61.47	63.45	65.14	66.55	67.66	68.35	68.21	68.31
node2vec ($p = 1, q = 0.5$)	58.93	62.60	64.11	65.36	66.03	67.38	67.93	68.26	68.99
SDNE	52.42	57.34	60.15	62.35	63.18	64.21	64.71	65.63	65.60
MNMF	54.76	58.82	60.43	61.66	62.74	63.23	63.46	63.45	64.77
ComE	59.11	62.46	64.38	65.45	65.98	67.38	67.49	67.92	67.89
S-DW	59.97	63.41	65.48	67.03	67.95	69.15	69.45	70.03	70.63
E-DW	58.69	62.37	64.01	65.46	66.16	66.85	66.79	67.05	67.05
S-n2v ($p = 2, q = 1$)	**60.66**	**64.43**	**66.63**	**68.23**	**68.92**	**70.52**	**70.41**	**70.62**	**71.60**
E-n2v ($p = 1, q = 0.5$)	60.07	63.81	65.52	66.69	67.64	69.21	69.62	69.40	70.71

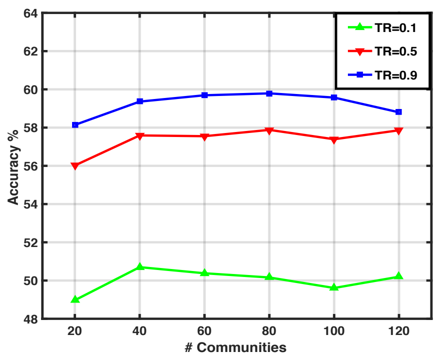
TABLE 5
Vertex classification results (%) on BlogCatalog (micro-precision).

%Training ratio	1%	2%	3%	4%	5%	6%	7%	8%	9%
DeepWalk	23.66	27.12	28.28	30.02	30.58	31.37	31.57	31.71	32.31
LINE	19.31	23.21	22.88	24.82	25.89	27.00	27.75	28.70	30.04
node2vec	24.47	27.83	29.11	30.61	30.87	31.05	31.50	31.44	31.96
SDNE	17.73	22.38	23.92	25.06	25.65	27.05	27.44	27.72	27.97
MNMF	19.26	21.08	22.29	23.99	25.24	25.97	26.31	26.58	27.16
ComE	22.67	27.43	28.49	29.79	30.34	31.04	31.32	31.58	32.15
S-DW	23.80	27.02	28.63	30.14	30.25	30.96	31.16	31.46	32.45
E-DW	24.93	**28.36**	29.28	**30.80**	**31.19**	31.65	31.72	**32.22**	**32.76**
S-n2v	24.95	27.88	29.17	30.24	30.95	**31.77**	**31.85**	32.12	32.34
E-n2v	**25.75**	28.29	**29.36**	30.54	31.13	31.36	31.67	31.99	32.60

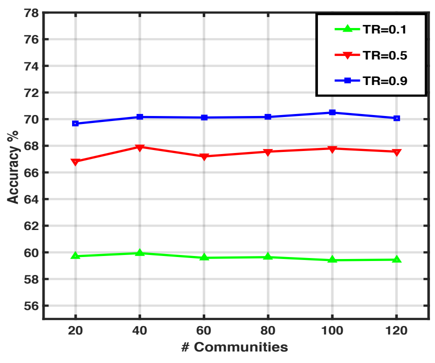
(1) The proposed CNRL model consistently and significantly outperforms all baseline methods on vertex classification. Specifically, Community-enhanced DeepWalk outperforms DeepWalk, as well as Community-enhanced node2vec outperforms node2vec.



(a) Classification accuracies on Cora



(b) Classification accuracies on Citeseer



(c) Classification accuracies on Wiki

Fig. 5. Parameter sensitivity on vertex classification. (TR denotes training ratio.)

It states the importance of incorporating community information and the flexibility of CNRL to various models. Moreover, with the consideration of community structure, CNRL is able to learn more meaningful and discriminative network representations and the learnt representations are suitable for predictive tasks.

(2) While MNMF performs poorly on Wiki and BlogCatalog, CNRL performs reliably on all datasets. Furthermore, CNRL achieves more than 4 points improvements than MNMF, although they both incorporate community information into NRL. It indicates that CNRL integrates community information more efficiently and has more stable performance than MNMF.

(3) With half less training data, CNRL still outperforms the baseline methods on different datasets. It demonstrates that CNRL can better handle the sparse situation.

To summarise, our proposed CNRL effectively encodes the community structure into vertex representations and achieves significant improvements compared with typical NRL methods on vertex classification. Besides, CNRL is flexible to various social networks, whether they are sparse or dense. Moreover, compared with typical NRL methods, it can obtain a better performance with much less labeled training data.

Parameter Sensitivity To verify the influence of the parameter in CNRL on vertex classification, e.g. the number of communities K, we conduct experiments on three datasets (e.g., Cora, Citeseer, and Wiki.) to analyze parameter sensitivity. Here, we employ the best-performed S-n2v and set the training ratio (TR) to 0.1, 0.5 and 0.9 respectively and vary the number of communities from 20 to 120. As shown in Fig. 5, our algorithm CNRL has a stable performance on vertex classification when the number of communities varies from 20 to 120. It indicates that our model can adapt to detect various numbers of communities during NRL. It also demonstrates the robustness of our model.

4.5 Link Prediction

Community modeling models should have the ability to predict links correctly. Therefore, we employ link prediction task to evaluate our proposed Community-enhanced NRL model.

Given a network, we randomly remove a portion of links as the test set and the rest as the training set. We utilize the observed links to learn vertex representations and employ these representations to measure the similarity between two vertices, which can be further applied to predict potential links between vertices.

We choose two kinds of baselines: link prediction methods and representation learning methods. All algorithms need to score the similarity between two vertices. Note that potentially connected vertices are expected to have higher similarity score than irrelevant ones.

We employ a standard link prediction metric, AUC [72], to evaluate baselines and our method. Given the similarity of all vertex pairs, AUC is the probability that a random unobserved link has higher similarity than a random nonexistent link. Assume that we draw n independent comparisons, the AUC value is

$$AUC = \frac{n_1 + 0.5n_2}{n} \tag{17}$$

where n_1 is the times that unobserved link has a higher score and n_2 is the times that they have an equal score.

In Fig. 6, we show the AUC values of link prediction on different datasets while removing 5% edges. Note that, we show the results of LINE-1st on BlogCatalog, as it outperforms LINE. From this table, we observe that:

(1) In most cases, NRL methods outperform traditional handcrafted link prediction methods. It proves that NRL methods are effective to encode network structure into real-valued representation vectors. In this case, our proposed CNRL model consistently outperforms other NRL methods on different datasets. The results demonstrate the reasonability and effectiveness of considering community structure again.

(2) For BlogCatalog, the average degree (i.e., 32.39) of vertices is much larger than other networks, which will benefit simple statistic-based methods, such as CN and RA. Nevertheless, as shown in Fig. 6, when the network turns sparse by removing 80% edges, the performance of these simple methods will badly decrease (25% around). On the contrary, CNRL only decreases about 5%. It indicates that CNRL is more robust to the sparsity issue.

(3) We propose two implementations of CNRL to assign community label for vertices, denoted as statistics-based CNRL (S-CNRL) and embedding-based CNRL (E-CNRL). Regardless of the proportions of removed links, S-CNRL outperforms E-CNRL on the citation networks of Cora and Citeseer, while E-CNRL performs better than S-CNRL on the web page network Wiki. It states the necessity and rationality of the proposed two instantiations of CNRL. It makes CNRL suitable to different kinds of networks.

Observations above demonstrate the effectiveness and robustness of CNRL for incorporating community structure into vertex representations. It achieves consistent improvements than other state-of-the-art NRL methods on link prediction task. Moreover,

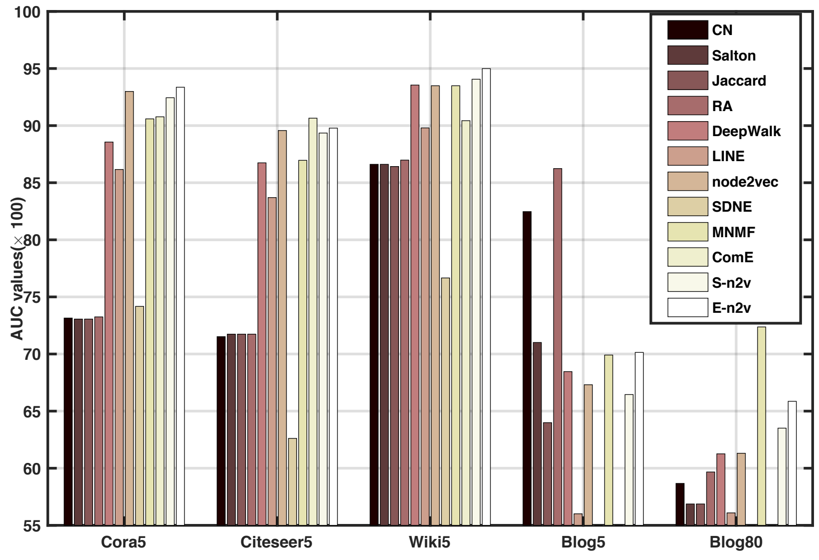
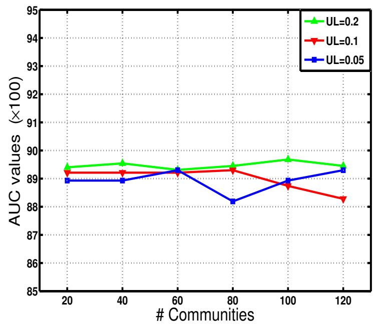
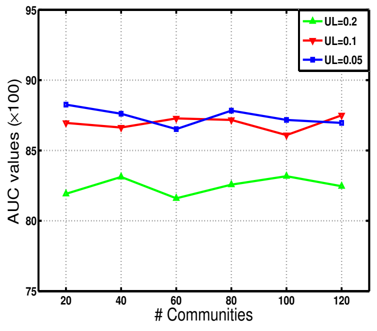


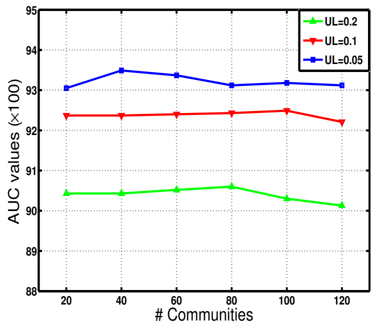
Fig. 6. Link prediction results when removing different amounts of links.



(a) AUC values on Cora



(b) AUC values on Citeseer



(c) AUC values on Wiki

Fig. 7. Parameter sensitivity on link prediction. (UL denotes unobserved links.)

with the two well-designed implementations, CNRL is flexible to various kinds of social networks.

Parameter Sensitivity We also perform experiments to analyze the influence of community size K on link prediction. Same to the operations in vertex classification, we also employ S-n2v and vary community size K from 20 to 120 and plot the AUC curves in Fig. 7. As shown in this figure, our algorithm CNRL has a stable performance when the community size changes from 20 to 120. To be specific, the AUC values range within 2% on all the three datasets, which demonstrates that the performance of our model is insensitive to the settings of the hyper-parameter K. It indicates the robustness and flexibility of our model.

4.6 Community Detection

As there is no ground-truth communities of these datasets, we use modified modularity [73] to evaluate the quality of detected communities. Since our models and BigCLAM obtain community distribution of each vertex, we simply set a threshold $\tau = 0.1$ to select communities of each vertex (i.e., keep communities whose probability is larger than τ). From Table 6, we observe that S-CNRL (S-DW or S-n2v) is comparable with other state-of-the-art community detection methods, while E-CNRL (E-DW or E-n2v) significantly outperforms these baselines. It states that the communities detected by CNRL are meaningful under the measurement of community quality. Moreover, it conforms to our assumptions about community assignment.

To summarize, all the results demonstrate the effectiveness and robustness of CNRL for incorporating community structure into vertex representations. It achieves consistent improvements comparing with other NRL methods on all network analysis tasks.

4.7 Scalability Test

To verify the efficiency of our model regarding the size of networks, we implement the best-performed S-n2v on a large-scale network dataset Youtube[2], which contains $1,138,499$ vertices and $2,990,443$ edges. Specifically, we randomly sample various proportions of vertices and test the training time of S-n2v on these sub-graphs. As shown in Fig. 9, S-n2v costs linear time with the number of vertices. This is mainly because most of the time cost is used for training LDA models, which is linear with the number of vertices as mentioned in the complexity analysis part. Therefore, how to accelerate the training speed of LDA models is expected to increase the efficiency of our model.

2. http://socialnetworks.mpi-sws.org/data-imc2007.html

TABLE 6
Community detection results.

Datasets	SCP	LC	MDL	BigCLAM	S-DW	E-DW	S-n2v	E-n2v
Cora	0.076	0.334	0.427	0.464	0.464	**1.440**	0.447	1.108
Citeseer	0.055	0.315	0.439	0.403	0.486	**1.861**	0.485	1.515
Wiki	0.063	0.322	0.300	0.286	0.291	**0.564**	0.260	0.564

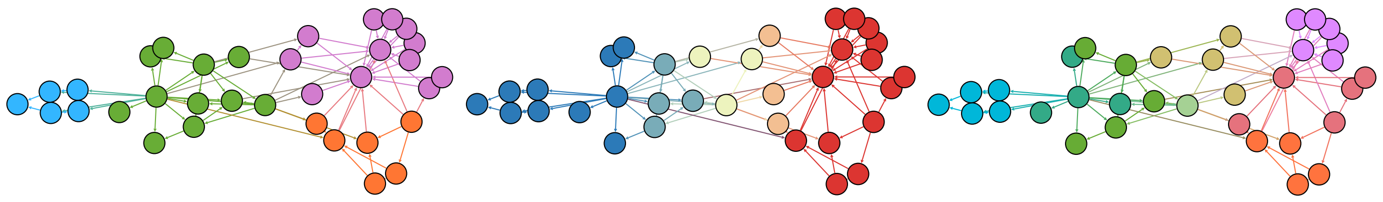


Fig. 8. Detected communities on Karate (Fast Unfolding, 2 communities by CNRL, 4 communities by CNRL).

TABLE 7
Top-5 nearest neighbors of the paper "Protein Secondary Structure Modelling with Probabilistic Networks" by DeepWalk and CNRL

CNRL	
Using Dirichlet Mixture Priors to Derive Hidden Markov Models for Protein Families	Neural Networks
Optimal Alignments in Linear Space using Automaton-derived Cost Functions	Neural Networks
Dirichlet Mixtures: A Method for Improving Detection of Weak but Significant Protein Sequence Homology	Neural Networks
Family-based Homology Detection via Pairwise Sequence Comparison	Neural Networks
The megaprior heuristic for discovering protein sequence patterns	Neural Networks
DeepWalk	
An Optimal Weighting Criterion of Case Indexing for Both Numeric and Symbolic Attributes	Case Based
Using Dirichlet Mixture Priors to Derive Hidden Markov Models for Protein Families	Neural Networks
On The State of Evolutionary Computation	Genetic Algorithms
Optimal Alignments in Linear Space using Automaton-derived Cost Functions	Neural Networks
On Biases in Estimating Multi-Valued Attributes	Rule Learning

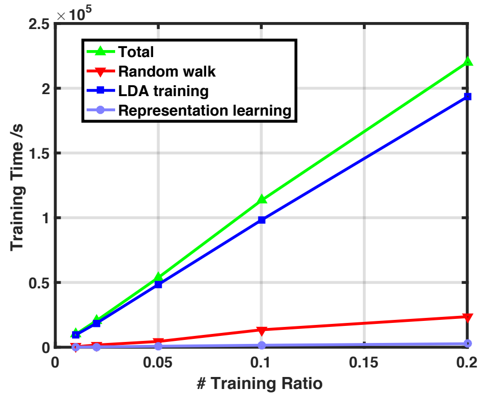


Fig. 9. Training time of S-n2v under various training ratios.

4.8 Visualizations of Detected Communities

For a more intuitive sense of detected communities, we visualize the detected overlapping communities by CNRL on a toy network named Zachary's Karate network [68] in Fig. 8. For comparison, we also show the detected non-overlapping communities by a typical community detection algorithm, Fast Unfolding [74]. Note that, we mark different communities with different colors, and use gradient color to represent the vertices belonging to multiple communities. From Fig. 8, we observe that:

(1) CNRL is able to detect community structure with multiple scales, rather than clustering or partitioning vertices into fixed communities. Both the 2-community version and 4-community one are straightforward and reasonable according to the network structure.

(2) CNRL is well versed dealing with the overlapping issues in community detection. It can accurately identify vertices on community boundaries and balance the weights of the communities they belong to.

Note that, in CNRL we assume the vertex sequences reserve global community patterns, and the communities are detected with vertex sequences. Results from Fig. 8 conform to our intuition and verify the assumption.

4.9 Case Study

To demonstrate the significance of CNRL and give an intuitive experience on communities, we conduct three representative case studies as follows.

4.9.1 Nearest neighbors

We provide a case in Cora to state the importance of considering community structure in NRL. The "paper" is titled as "Protein

TABLE 8
Community assignments and representative vertices.

Community 1 (weight = 0.56)	
Learning to Act using Real-Time Dynamic Programming	Reinforcement Learning
Generalized Markov Decision Processes: Dynamic-programming and Reinforcement-learning Algorithms	Reinforcement Learning
On the Convergence of Stochastic Iterative Dynamic Programming Algorithms	Reinforcement Learning
Community 2 (weight = 0.20)	
The Structure-Mapping Engine: Algorithm and Examples	Case Based
Case-based reasoning: Foundational issues, methodological variations, and system approaches	Case Based
Concept Learning and Heuristic Classification in Weak-Theory Domains	Case Based
Community 3 (weight = 0.12)	
Learning to Predict by the Methods of Temporal Differences	Reinforcement Learning
Generalization in Reinforcement Learning: Safely Approximating the Value Function	Reinforcement Learning
Exploration and Model Building in Mobile Robot Domains	Reinforcement Learning

Secondary Structure Modelling with Probabilistic Networks" and belongs to the research field of "Neural Networks". With learnt representation vectors by DeepWalk and S-CNRL, we measure the distance between vertices with cosine similarity and give the top-5 nearest neighbors in Table 7.

From Table 7, the most straightforward observation is that CNRL gives 5 nearest neighbors with the same labels, while DeepWalk only gives 2. It's reasonable as vertices in a community attempt to share the same attributes. The attributes are always hidden in the communities, which are utilized by CNRL. More detailed observations can be achieved by analyzing their topics. The provided "paper" is related to topics including "protein","structure" and "probabilistic networks". Most of the neighbors found by DeepWalk are unrelated to these topics. In contrast, most of the neighbors found by CNRL are closely correlated with them. More specifically, the nearest neighbors found by CNRL can be roughly assigned into "protein" related and "probabilistic network" related categories, which indicates the existence of latent correlations with community structure.

4.9.2 Community Assignments

Compared with DeepWalk, CNRL can learn not only the community-enhanced vertex representations but also the community assignments. We provide a case in Cora and its community assignments in Table 8. The selected "paper" is titled as "Using a Case Base of Surfaces to Speed-Up Reinforcement Learning" and belongs to the field of "Reinforcement Learning". For each community, we follow Eq. (6) to select representative vertices.

From this table, we observe that each community has its own characteristics. For example, community 1 is related to "Dynamic Programming", which is a sub-field in "Reinforcement Learning". Community 2 is relevant to "Cased Based" research, and community 3 concerns more about the learning and modeling methods in "Reinforcement Learning". According to the title of the selected vertex, we find that it is actually relevant to all these communities and the weights can reflect its relevance to the communities.

4.9.3 Global Community Patterns

In this part, we attempt to make clear the insights of community patterns from an overall perspective.

We train S-CNRL on Cora when $K = 20$, and select the most related vertices according to whether $\Pr(v|c) > \eta$. Here, the threshold η is set to 0.005. Afterwards, we identify the most-frequent words in titles for each community. We ignore "Machine" and "Learning" as Cora is a machine learning paper set and the titles of most papers contain these words. Due to space limitations, we only list 6 typical communities in Table 9.

TABLE 9
Representative words of communities in Cora

Community ID	Words and Frequencies
0	Models:13 Hidden:11 Markov:10
6	Reasoning:13 Case-Based:13 Knowledge:7
8	Genetic:23 Programming:16 Evolution:9
13	Boosting:6 Bayesian:6 Classifiers:6
15	Neural:14 Networks:11 Constructive:7
19	Reinforcement:21 Markov:8 Decision:8

From this table, we get a primary experience of the division in machine learning filed. These communities are discriminative and corresponding to "Hidden Markov Models", "Case-based Reasoning", "Genetic Programming", "Neural Networks" and "Reinforcement Learning" respectively. It demonstrates that CNRL is effective in detecting high-quality communities.

5 Conclusion and Future Work

In this paper, we propose a novel community enhancement mechanism for NRL. By taking full consideration of the critical community information, the proposed method jointly embeds local vertex characteristics and global community patterns into vertex representations, as well as simultaneously detecting community structure. Moreover, CNRL can be easily implemented on conventional random walk based NRL models, e.g. DeepWalk and node2vec. It achieves significant improvements compared with existing state-of-the-art NRL algorithms when applying the learnt representations to network analysis tasks. Besides, CNRL can effectively detect overlapping communities on multiple scales.

In future, we will explore the following directions:

(1) In this work, we employ topic modeling methods on random walk sequences for community detection inspired by the

analogy between natural language and networks. However, the correlation between topics and communities still lacks theoretical demonstration. Therefore, we aim to prove this analogy mathematically through matrix factorization.

(2) We seek to investigate the extensibility of our model on incorporating heterogeneous information in social networks, such as text contents and attributes. Our method is expected to be flexible to incorporate the heterogeneous information comprehensively.

(3) Another intriguing direction would be semi-supervised learning of our model. We will adapt the representation learning for specific tasks such as vertex classification and take label information of training set into account to enhance the quality of vertex representations for prediction.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China (NSFC No. 61772302, 61532010, 61661146007). This research is also part of the NExT++ project, supported by the National Research Foundation, Prime Ministers Office, Singapore under its IRC@Singapore Funding Initiative. Tu is also supported by China Postdoctoral Innovative Talent Support Programme.

REFERENCES

[1] A. Shepitsen, J. Gemmell, B. Mobasher, and R. Burke, "Personalized recommendation in social tagging systems using hierarchical clustering," in *Proceedings of RecSys*, 2008.

[2] N. A. Heard, D. J. Weston, K. Platanioti, D. J. Hand *et al.*, "Bayesian anomaly detection methods for social networks," *AOAS*, 2010.

[3] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *JASIST*, 2007.

[4] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of KDD*, 2014, pp. 701–710.

[5] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of WWW*, 2015, pp. 1067–1077.

[6] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proceedings of KDD*, 2016.

[7] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of CIKM*, 2015.

[8] M. E. Newman, "Modularity and community structure in networks," *PNAS*, 2006.

[9] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," in *Proceedings of ICDM*, 2013.

[10] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of NIPS*, 2013, pp. 3111–3119.

[11] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proceedings of AAAI*, 2017.

[12] S. Cavallari, V. W. Zheng, H. Cai, K. C.-C. Chang, and E. Cambria, "Learning community embedding with community detection and node embedding on graphs," in *Proceedings of CIKM*, 2017.

[13] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel, "You are who you know:inferring user profiles in online social networks," in *Proceedings of WSDM*, 2010, pp. 251–260.

[14] R. Li, C. Wang, and K. C. C. Chang, "User profiling in an ego network: co-profiling attributes and relationships," in *Proceedings of WWW*, 2014, pp. 819–830.

[15] L. Akoglu, M. Mcglohon, and C. Faloutsos, "Anomaly detection in large graphs," *Cmu-Cs-Technical Report*, 2010.

[16] R. Hassanzadeh, R. Nayak, and D. Stebila, "Analyzing the effectiveness of graph metrics for anomaly detection in online social networks," in *Proceedings of WISE*, 2012, pp. 624–630.

[17] M. Fire, G. Katz, and Y. Elovici, "Strangers intrusion detection - detecting spammers and fake profiles in social networks based on topology anomalies," *Computer Methods in Biomechanics & Biomedical Engineering*, vol. 11, no. 11, pp. 83–84, 2012.

[18] Z. Chen, W. Hendrix, and N. F. Samatova, "Community-based anomaly detection in evolutionary networks," *Journal of Intelligent Information Systems*, vol. 39, no. 1, pp. 59–85, 2012.

[19] S. Rosenthal and K. McKeown, "Age prediction in blogs: A study of style, content, and online behavior in pre-and post-social media generations," in *Proceedings of ACL*, 2011, pp. 763–772.

[20] J. D. Burger, J. Henderson, G. Kim, and G. Zarrella, "Discriminating gender on twitter," in *Proceedings of EMNLP*, 2011, pp. 1301–1309.

[21] C. Tu, Z. Liu, H. Luan, and M. Sun, "Prism: Profession identification in social media," *ACM TIST*, vol. 8, no. 6, p. 81, 2017.

[22] H. A. Schwartz, J. C. Eichstaedt, M. L. Kern, L. Dziurzynski, S. M. Ramones, M. Agrawal, A. Shah, M. Kosinski, D. Stillwell, M. E. Seligman *et al.*, "Personality, gender, and age in the language of social media: The open-vocabulary approach," *PloS one*, vol. 8, no. 9, p. e73791, 2013.

[23] C. Tu, W. Zhang, Z. Liu, and M. Sun, "Max-margin deepwalk: Discriminative learning of network representation," in *Proceedings of IJCAI*, 2016.

[24] J. Li, J. Zhu, and B. Zhang, "Discriminative deep random walk for network classification," in *Proceedings of ACL*, 2016.

[25] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *Proceedings of ICML*, 2016, pp. 40–48.

[26] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of ICLR*, 2017.

[27] M. E. Newman, "Clustering and preferential attachment in growing networks," *Physical Review E*, 2001.

[28] G. Salton and M. J. McGill, "Introduction to modern information retrieval," 1986.

[29] T. Zhou, L. Lü, and Y.-C. Zhang, "Predicting missing links via local information," *EPJ B*, 2009.

[30] H. Kashima, T. Kato, Y. Yamanishi, M. Sugiyama, and K. Tsuda, "Link propagation: A fast semi-supervised learning algorithm for link prediction." in *SDM*, 2009.

[31] R. Raymond and H. Kashima, "Fast and scalable algorithms for semi-supervised link prediction on static and dynamic graphs," in *Machine Learning and Knowledge Discovery in Databases*, 2010, pp. 131–147.

[32] A. K. Menon and C. Elkan, "Link prediction via matrix factorization," in *Machine Learning and Knowledge Discovery in Databases*, 2011, pp. 437–452.

[33] D. M. Dunlavy, T. G. Kolda, and E. Acar, "Temporal link prediction using matrix and tensor factorizations," *ACM TKDD*, vol. 5, no. 2, p. 10, 2011.

[34] S. Soundarajan and J. Hopcroft, "Using community information to improve the precision of link prediction methods," in *Proceedings of WWW*, 2012.

[35] N. Z. Gong, A. Talwalkar, L. Mackey, L. Huang, E. C. R. Shin, E. Stefanov, E. R. Shi, and D. Song, "Joint link prediction and attribute inference using a social-attribute network," *TIST*, 2014.

[36] B. Yang, D. Liu, and J. Liu, "Discovering communities from social networks: Methodologies and applications," in *Handbook of social network technologies and applications*, 2010, pp. 331–346.

[37] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3, pp. 75–174, 2010.

[38] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell system technical journal*, vol. 49, no. 2, pp. 291–307, 1970.

[39] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.

[40] A. Pothen, H. D. Simon, and K.-P. Liou, "Partitioning sparse matrices with eigenvectors of graphs," *SIAM journal on matrix analysis and applications*, vol. 11, no. 3, pp. 430–452, 1990.

[41] K. Nowicki and T. A. B. Snijders, "Estimation and prediction for stochastic blockstructures," *JASA*, vol. 96, no. 455, pp. 1077–1087, 2001.

[42] T. P. Peixoto, "Model selection and hypothesis testing for large-scale network models with overlapping groups," *Physical Review X*, vol. 5, no. 1, p. 011033, 2015.

[43] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, 2005.

[44] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann, "Link communities reveal multiscale complexity in networks," *Nature*, 2010.

[45] F. Wang, T. Li, X. Wang, S. Zhu, and C. Ding, "Community discovery using nonnegative matrix factorization," *Data Mining and Knowledge Discovery*, 2011.

[46] J. Yang and J. Leskovec, "Community-affiliation graph model for overlapping network community detection," in *Proceedings of ICDM*, 2012.

[47] J. Xie, S. Kelley, and B. K. Szymanski, "Overlapping community detection in networks: The state-of-the-art and comparative study," *CSUR*, 2013.
[48] J. Yang and J. Leskovec, "Overlapping community detection at scale: a nonnegative matrix factorization approach," in *Proceedings of WSDM*, 2013.
[49] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of NIPS*, 2012.
[50] L. Tang and H. Liu, "Relational learning via latent social dimensions," in *Proceedings of KDD*, 2009.
[51] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *Proceedings of IJCAI*, 2015.
[52] L. F. R. Ribeiro, P. H. P. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proceedings of KDD*, 2017.
[53] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of KDD*, 2016.
[54] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proceedings of AAAI*, 2016, pp. 1145–1152.
[55] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of NIPS*, 2017.
[56] H. Li, H. Wang, Z. Yang, and M. Odagaki, "Variation autoencoder based network representation learning for classification," in *Proceedings of ACL*, 2017.
[57] C. Tu, H. Liu, Z. Liu, and M. Sun, "Cane: Context-aware network embedding for relation modeling," in *Proceedings of ACL*, 2017.
[58] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proceedings of KDD*, 2015, pp. 119–128.
[59] C. Tu, Z. Zhang, Z. Liu, and M. Sun, "Transnet: Translation-based network representation learning for social relation extraction," in *Proceedings of IJCAI*, 2017.
[60] J. Qiu, Y. Dong, H. Ma, J. Li, K. Wang, and J. Tang, "Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec," in *Proceedings of WSDM*, 2018, pp. 459–467.
[61] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of ICIR*, 2013.
[62] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, 2001.
[63] T. L. Griffiths and M. Steyvers, "Finding scientific topics," *PNAS*, 2004.
[64] Y. Wang, H. Bai, M. Stanton, W.-Y. Chen, and E. Y. Chang, "Plda: Parallel latent dirichlet allocation for large-scale applications," in *Proceedings of AAIM*, 2009, pp. 301–314.
[65] Z. Liu, Y. Zhang, E. Y. Chang, and M. Sun, "Plda+: Parallel latent dirichlet allocation with data placement and pipeline processing," *ACM TIST*, vol. 2, no. 3, p. 26, 2011.
[66] A. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval Journal*, 2000.
[67] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, 2008.
[68] W. W. Zachary, "An information flow model for conflict and fission in small groups," *JAR*, 1977.
[69] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A*, 2011.
[70] J. M. Kumpula, M. Kivelä, K. Kaski, and J. Saramäki, "Sequential algorithm for fast clique percolation," *Physical Review E*, 2008.
[71] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *JMLR*, 2008.
[72] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve," *Radiology*, 1982.
[73] H. Zhang, I. King, and M. R. Lyu, "Incorporating implicit link preference into overlapping community detection," in *Proceedings of AAAI*, 2015.
[74] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *JSTAT*, 2008.

Cunchao Tu is a postdoc of the Department of Computer Science and Technology, Tsinghua University. He got his BEng degree in 2013 from the Department of Computer Science and Technology, Tsinghua University. His research interests are natural language processing and social computing. He has published several papers in international conferences and journals including ACL, IJCAI, EMNLP, COLING, and ACM TIST.

Xiangkai Zeng is an undergraduate student in the School of Computer Science and Engineering, Beihang University, Beijing, China. His research interests are natural language processing and deep learning.

Hao Wang is a PhD student in the School of Physical and Mathematical Sciences, Nanyang Technological University. He got his BEng degree in 2017 from the Department of Computer Science and Technology, Tsinghua University. His research interests include algorithmic game theory, mechanism design, and social network analysis.

Zhengyan Zhang is an undergraduate student of the Department of Computer Science and Technology, Tsinghua University. His research interests include natural language processing and social computing.

Zhiyuan Liu is an associate professor of the Department of Computer Science and Technology, Tsinghua University. He got his BEng degree in 2006 and his Ph.D. in 2011 from the Department of Computer Science and Technology, Tsinghua University. His research interests are natural language processing and social computation. He has published over 40 papers in international journals and conferences including ACM Transactions, IJCAI, AAAI, ACL and EMNLP.

Maosong Sun is a professor of the Department of Computer Science and Technology, Tsinghua University. He got his BEng degree in 1986 and MEng degree in 1988 from Department of Computer Science and Technology, Tsinghua University, and got his Ph.D. degree in 2004 from Department of Chinese, Translation, and Linguistics, City University of Hong Kong. His research interests include natural language processing, Chinese computing, Web intelligence, and computational social sciences. He has published over 150 papers in academic journals and international conferences in the above fields. He serves as a vice president of the Chinese Information Processing Society, the council member of China Computer Federation, the director of Massive Online Education Research Center of Tsinghua University, and the Editor-in-Chief of the Journal of Chinese Information Processing.

Bo Zhang is a researcher of the Search Product Center, WeChat Search Application Department, Tencent Inc., China. He got his BEng degree in 2009 from School of Computer Science, Xidian University, China, and got his Master Degree in 2012 from School of Computer Science and Technology, Zhejiang University, China.

Leyu Lin is a researcher of the Search Product Center, WeChat Search Application Department, Tencent Inc., China. He got his BEng degree in 2004 from Department of Computer Science and Technology, Tianjin Normal University, China.