Information Sciences 542 (2021) 343–356

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

HNS: Hierarchical negative sampling for network representation learning

Junyang Chen [a], Zhiguo Gong [a,*], Wei Wang [a,*], Weiwen Liu [b]

[a] Department of Computer Information Science, University of Macau, Macau, China
[b] Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China

ARTICLE INFO

Article history:
Received 11 October 2019
Received in revised form 5 July 2020
Accepted 8 July 2020
Available online 22 July 2020

Keywords:
Hierarchical negative sampling
Network representation learning
Network embeddings

ABSTRACT

Network representation learning (NRL) aims at modeling network graph by encoding vertices and edges into a low-dimensional space. These learned representations can be used for subsequent applications, such as vertex classification and link prediction. Negative Sampling (NS) is the most widely used method for boosting the performance of NRL. However, most of the existing work only randomly draws negative samples based on vertex frequencies, i.e., the vertices with higher frequency are more likely to be drawn, which ignores the situation that the sampled one may not be a true negative sample, thus, lead to undesirable embeddings. In this paper, we propose a new negative sampling method, called Hierarchical Negative Sampling (HNS), which is able to model the latent structures of vertices and learn the relations among them. During sampling, HNS can draw more appropriate negative samples and thereby obtain better performance on network embeddings. Firstly, we theoretically demonstrate the superiority of HNS over NS. And then we use experimental results to show that our proposed method outperforms the state-of-the-art models on vertex classification tasks at different training scales in real-world networks.

1. Introduction

Network representation learning (NRL), i.e., network embeddings, can map the semantic similarity of vertices into a low-dimensional space where the similar vertices presented in the network are assigned to the nearby areas in the space [8]. The learned representations are significant for the subsequent applications, such as link prediction [15], vertex classification [25], and community detection [35]. The vital part of network embeddings is to discover the neighbors of vertices and learn the representations which can well preserve the proximity in networks. For this objective, DeepWalk [25] explores neighbor relations by using depth-first search (DFS) and Line [29] utilizes breadth-first search (BFS) to discover vertex proximity. To obtain more complete exploration, Node2vec [15] employs both DFS and BFS on networks. After searching out neighbors of vertices, most of NRL methods adopt Skip-Gram [21], a language model that maximizing the probability of word co-occurrence (corresponding to vertex neighbors in graphs) within a sliding window, to learn vertex representations. Due to the computational complexity of the objective in Skip-Gram, these methods mainly use negative sampling (NS) [22] to approximate a variant objective of Skip-Gram. Though other optimization methods such as Hierarchical Softmax

* Corresponding authors.
E-mail addresses: yb77403@umac.mo (J. Chen), fstzgg@umac.mo (Z. Gong), wwang@um.edu.mo (W. Wang), wwliu@cse.cuhk.edu.hk (W. Liu).

https://doi.org/10.1016/j.ins.2020.07.015
0020-0255/

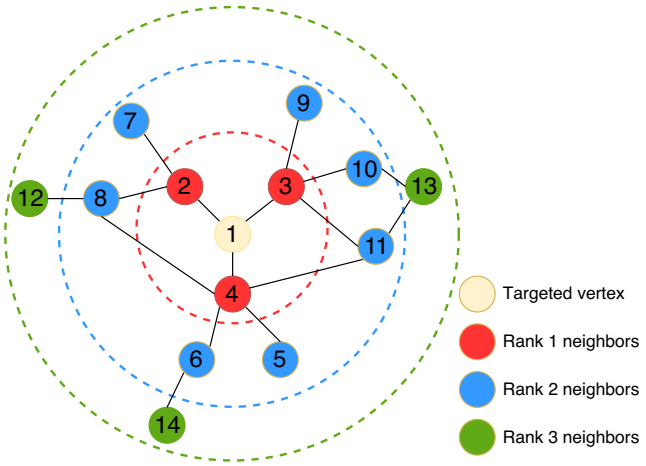


Fig. 1. An example of the neighbors for the targeted vertex 1. From center to periphery, vertices in the first circle are the rank 1 neighbors and marked in red. Vertices located in the second and the third circles are the rank 2 and rank 3 neighbors and marked in blue and green, respectively.

[24] and Noise Contrastive Estimation [17] are proposed, the experimental results in [29,15,26] show that NS can generally achieve better performance in large-scale datasets. The major idea of NS is to encourage a target vertex to be close to its neighbors and in the meanwhile be far from its negative samples.

Nevertheless, existing NS methods draw negative samples based on the frequencies of vertices and ignore if they are really negative to the given vertex. In other words, for a target vertex, its adjacent vertices may be drawn as the negative samples, which leads to undesirable embeddings. To address this problem, R-NS [1] defines Distance Negative Sampler which chooses negative samples for each vertex u from the set $V - \{N(u) \cup u\}$, where V is the set of vertices and $N(u)$ is the neighbors of u. To give a more clear explanation, an illustration is presented in Fig. 1. For the targeted vertex 1, R-NS aims to choose negative samples except from rank 1 neighbors[1] (e.g., the direct neighbor vertices 2, 3, and 4 marked in red). However, it is not sufficient to merely take only rank 1 neighbors into consideration. For example, in the rank 2 neighbors, vertex 11 is associated with vertices 3 and 4 which are highly relevant to the targeted vertex 1. Therefore, it may lead to inferior embeddings if we choose vertex 11 as the negative sample of vertex 1. Nonetheless, to the best of our knowledge, there are no efforts have been devoted to considering the rank N neighbor information when sampling negative samples.

In addition, some recent attempts are proposed for improving NS. Chen et al. [7] propose an adaptive sampler in the Skip-Gram model which can increase the precision of the estimation in NS by ranking all words in the vocabulary every few iterations of the stochastic gradient descent. Wang et al. [37] propose a GAN-based framework that incorporates Generative Adversarial Networks to obtain high-quality negative samples with a generator and learns the embeddings of vertices with a discriminator. To reduce the training time on GAN, Zhang et al. [49] propose NSCaching which essentially is a cache mechanism for storing informative negative samples. However, all these methods draw negative samples without considering the neighbors' information. In brief, for the targeted vertex 1 as mentioned before, they may still choose vertex 11 as its negative sample.

Moreover, it is always difficult to determine an optimal rank number for reserving neighbor areas where the vertices are not allowed to be chosen as negative samples for the targeted vertices. For example, if we only use rank 1 neighbors, some vertices in rank 2 or 3 which are highly related to the targeted vertex may be selected as the negative ones. On the contrary, if we preserve rank 2 and 3 neighbors, irrelevant vertices may be retained and lead to undesirable embeddings.

Motivated by the above discussions, in this paper we propose a new negative sampling method, called Hierarchical Negative Sampling (HNS), to model the rank N neighbor information. Rather than computing the frequencies of vertices or setting a hard threshold for reserving neighbor area, HNS adaptively discovers the latent community structures of vertices and computes the probabilities of their being negative samples. More specifically, for each targeted vertex, HNS firstly assigns it to a community, then formulates a reversed community-vertex distribution to draw negative samples for optimizing its objective function. By modeling the neighbor proximity information into the hierarchical community structures, HNS can obtain more appropriate negative samples and learn better representations of vertices.

Contributions of our work are summarized as follows:

- We propose a new negative sampling method, HNS, which can explore the latent structures of vertices and exploit the rank N neighbor information of targeted vertices when selecting negative samples. To the best of our knowledge, this paper is the first attempt to investigate drawing negative samples with modeling neighbor information.

[1] Notice that 'neighbor rank' is identical to 'neighbor order' in this paper.

- The reversed community-vertex distributions learned by HNS can be regarded as the probabilities of vertices to be drawn as negative samples, which avoids setting rank numbers of reserved neighbors for each vertex manually.
- We conduct experiments on three real-world datasets including citation networks and language networks. Comparing with the state-of-the-art models, experimental results demonstrate that HNS can achieve better performance in vertex classification tasks at different training scales.

The code and datasets are released at https://github.com/junyachen/HNS. The rest of this paper is organized as follows. In Section 2, we give preliminaries of notations and related models. Section 3 presents our proposed model and the parameter inference method. We discuss experimental results in Section 4, and introduce the related work in Section 5. Section 6 concludes our work.

2. Preliminaries

Here, we introduce the general components for training network embeddings along with necessary notations and formalizations.

2.1. Random walks

Given a network $G = (V, E)$, where V is the set of vertices and $E \subseteq V \times V$ denotes the set of edges, random walks [25] is performed to form a set of walk sequences $S = \{s_1, \ldots, s_M\}$ and each sequence s consists a set of word frequencies $\{v_1, \ldots, v_N\}$. M denotes the total number of walk sequences and N denotes the size of unique vertices.

2.2. Skip-Gram model

Skip-Gram [22] model is one hidden layer neural network which is firstly introduced for word embeddings. By treating vertices as words and sequences as documents, the Skip-Gram model has been successfully extended to network embeddings [25]. The objective of Skip-Gram model is to maximize the average log probability of predicting the context vertices $\{v_{i-t}, \ldots, v_{i+t}\} \setminus v_i$ within window t for each vertex v_i in sequence s, which is shown as follows:

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

where v_j is the context vertex of vertex v_i, $|s|$ denotes the number of vertices in sequence s, and probability $P(v_j|v_i)$ is a softmax unit:

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

where $\mathbf{v} \in \mathbb{R}^d$ denotes the learned low-dimensional vector and d represents the dimension of representation space.

2.3. Negative Sampling

As shown in Eq. (2), each softmax term requires a summation over all changing vectors. Therefore, the objective of Skip-Gram is computationally infeasible in large-scale networks. Negative sampling (NS) [22] simplifies this objective but retains its quality. Given targeted vertex v_i and its context vertex v_j, NS approximates it as:

$$\mathcal{L}_{NS} = \log(\sigma(\mathbf{v}'_j \cdot \mathbf{v}_i)) + \sum_{k=1}^{K} \mathbb{E}_{v_k \sim P_{NS}(v)} \log(\sigma(-\mathbf{v}'_k \cdot \mathbf{v}_i)), \tag{3}$$

where $\sigma(x) = 1/(1 + \exp(-x))$, $P_{NS}(v)$ is a negative sample distribution, v_k is a sample drawn from $P_{NS}(v)$, and K is the number of negative samples for the estimation.

3. Hierarchical negative sampling

In this section, we first formally explain the problem of NS sampling for estimating its objective, which also exists in the state-of-the-art models such as NSCaching [49], R-NS [1], and SGA [7]. Then, we introduce our proposed model HNS. Lastly, we present how HNS can alleviate this problem.

3.1. Sampling from relevant vertex problem

As introduced in [22], NS models $P_{NS}(v)$ as a uniform distribution proportional to d_v^{β}, where d_v is the frequency of vertex v in all sequences, β is an empirical degree power usually set to 3/4. For each vertex, NS uniformly draws negative samples from a fixed distribution $P_{NS}(v)$, which is formulated as:

$$P_{NS}(v) = \frac{d_v^{\beta}}{\sum_{n \in V} d_n^{\beta}}. \tag{4}$$

Therefore, vertices with higher frequencies are more likely to be drawn as the negative samples for the targeted vertex. This strategy may induce worse embedding performance, as the example mentioned in the introduction, when vertices with high frequencies are located in rank 2 or 3 neighbor areas of the targeted vertex.

From Eq. (4), we can see that $P_{NS}(v)$ is a frequency-based fixed distribution from scratch, it thereby cannot model the latent structures of relevant vertex information and select negative samples properly. For the existing state-of-the-art models, e.g., [37,3,49,7], they are proposed to generate high-quality negative samples to improve NS by dynamically selecting vertices with large gradient magnitudes, i.e., $\sigma(-\mathbf{v}_k' \cdot \mathbf{v}_i)$ in Eq. (3), as the negative samples. However, their methods have high variance because relevant vertices from rank N neighbors may be regarded as negative samples in the initial state, and these reinforcement-based gradient methods will induce error propagation and lead to unstable performance [43]. Besides, other improved models, such as R-NS [1], as mentioned in the introduction, only consider rank 1 neighbors which is not sufficient for exploring the correlations among vertices.

3.2. Modeling neighbor proximity information

After performing random walks on a network, we can get walk sequences. As the analogy between topics in documents and communities in networks has been verified by [25,35], we can model neighbor proximity information into latent community structures. Without loss of generality, we follow the analogical assumptions of the classical topic model [2]: (1) Each walk sequence is an admixture of communities; (2) Each community has vertex preferences. Although community information has been explored by some NRL models [46,41,4,35], they are not designed for negative sample selection. Moreover, in the real-world, the community number is unknown in unsupervised learning. These methods thereby cannot handle well the dynamic changes of community discovery with a predefined community number. In the proposed HNS, we employ Hierarchical Dirichlet Processes (HDP) [32] to explore the latent community structures, which provides a nonparametric method for inferring the number of communities. The graphical model of HNS is as shown in Fig. 2 and the generative process is given as below:

1. Sample a global based distribution over all existing communities $C, G_0|\gamma, C \sim Dir(\gamma/C)$,
2. For each community $c \in \{1, 2, \ldots, C\}$:
   (a) Sample a distribution over vertices, $\boldsymbol{\varphi}_c|\beta \sim Dir(\beta)$,
3. For each walk sequence $s \in \{s_1, s_2, \ldots, s_M\}$:
   (a) Sample a community proportion, $\boldsymbol{\theta}_s|\alpha, G_0 \sim DP(\alpha, G_0)$,
   (b) For each vertex $v \in \{v_1, v_2, \ldots, v_N\}$:
      i. Draw a community assignment from the proportion, $z_{s,v}|\boldsymbol{\theta}_s \sim Multinomial(\boldsymbol{\theta}_s)$,
      ii. Sample a vertex, $v|z_{s,v}, \{\boldsymbol{\varphi}_c\}_{c=1}^{C} \sim Multinomial(\boldsymbol{\varphi}_{z_{s,v}})$,

where C denotes discovered communities and $C = 1$ in initialization, γ and β are the Dirichlet priors [2], DP indicates the Dirichlet process [31], and α represents the probability of vertex v choosing a new community. As $C \to \infty$, HNS is a HDP-based model which allows the community number to be automatically determined along the processing.

3.3. Training framework of HNS

The training framework of our proposed model is demonstrated in Fig. 3. To be specific, for each walk sequence, we firstly use a sliding window to generate positive samples, e.g., vertex 4 is the target and its positive vertex pairs are (4, 3) and (4, 5). Then, we follow the generative process as mentioned in Section 3.2 to sample a community assignment of vertex 4 and draw its negative samples by the reversed community-vertex distribution (the details of the inference process and formulations will be introduced in the following section). Finally, we incorporate the NS method to learn vertex representations.

3.4. Model inference

We adopt Gibbs sampling [14] in the graphical model inference. As shown in Fig. 2, vertex v is observed and its community assignment $z_{s,v}$ is latent. Because the conjugate priors of Dirichlet are used, θ_s and φ_c can be integrated out. Then we can sample $z_{s,v}$ from a conditional distribution $P(z_{s,v} = c|\mathbf{z}_{s,\neg v}, \mathbf{v}_{c,\neg s}, \alpha, \gamma, \beta)$, which is the probability of vertex v choosing community c given the information of other vertex community assignments in the sequence s except for current vertex v (i.e., $\mathbf{z}_{s,\neg v}$)

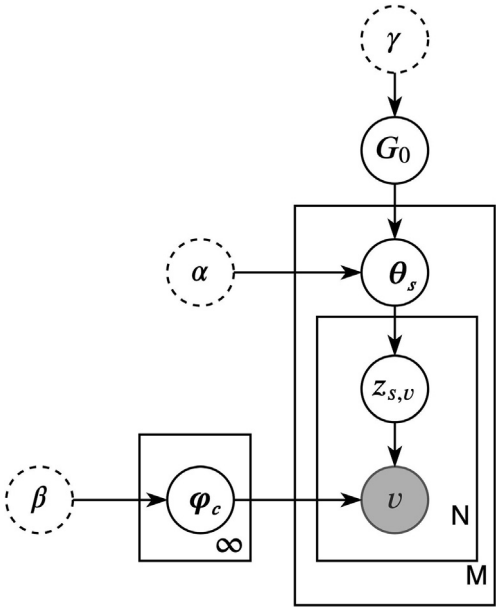


Fig. 2. The graphical model of HNS.

and the vertex community assignments in the community c except for current sequence s (i.e., $\mathbf{v}_{c,\neg s}$). In general, there are two cases for the sampled community c: choosing an existing community or a new one. The details are as follows:

3.4.1. Choosing an existing community

The probability of vertex v choosing existing communities C given the information of other sequences and their vertex assignments is computed as:

$$P(z_{s,v}=c|\mathbf{z}_{s,\neg v},\mathbf{v}_{c,\neg s},\alpha,\gamma,\beta)\propto P(z_{s,v}=c|\mathbf{z}_{s,\neg v},\alpha,\gamma)\ P(v|z_{s,v}=c,\mathbf{v}_{c,\neg s},\beta)=\frac{n^{c}_{s,\neg v}+\gamma/C}{\sum_{t=1}^{C}n^{t}_{s,\neg v}+\gamma+\alpha}\cdot\frac{n^{v}_{c,\neg s}+\beta}{\sum_{t=1}^{N}n^{t}_{c,\neg s}+N\beta}, \tag{5}$$

where $n^{c}_{s,\neg v}$ denotes the number of vertices assigned to community c in sequence s except for current vertex v, and $n^{v}_{c,\neg s}$ denotes the number of vertex v assigned to community c except for current sequence s. In essence, the first part of the mul-

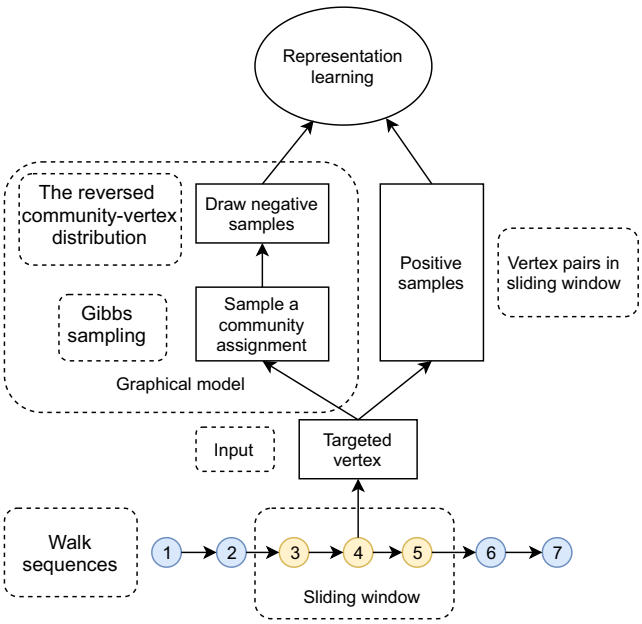


Fig. 3. The training framework of HNS.

tipliers in Eq. (5) means that vertex v tends to choose a community where more vertices in the same sequence are assigned to. The second part indicates that vertex v prefers to choose a community with higher vertex co-occurrences.

3.4.2. Choosing a new community

We use $C+1$ to represent a new community and formulate the conditional probability of vertex v choosing a new community as follows:

$$P(z_{s,v}=C+1|\mathbf{z}_{s,\neg v},\mathbf{v}_{c,\neg s},\alpha,\gamma,\beta)\propto P(z_{s,v}=C+1|\mathbf{z}_{s,\neg v},\alpha,\gamma)\ P(v|z_{s,v}=C+1,\beta)=\frac{\alpha}{\gamma+\alpha}\cdot\frac{1}{N}, \tag{6}$$

where the left part of the multipliers represents the potential probability of vertex v belonging to a new community, and the second part is the pseudo number of vertex occurrences in the new community. When α is larger, the vertex is more likely to choose a new community. In our experiments, we find that the classification performance of the proposed HNS is robust to α (The details are in Section 4.6).

3.4.3. Drawing negative samples

Recalling to the training framework as demonstrated in Fig. 3, we firstly sample a community assignment c for the targeted vertex v_t with Eqs. (5) and (6). Then, we can draw negative samples from the assigned community. The probabilities of other vertices being regarded as negative samples in community c are defined as follows:

$$P_{neg}(v)=1-P(v|z_{s,v_t}=c,\mathbf{v}_{c,\neg s},\beta)=1-\frac{n^{v}_{c,\neg s}+\beta}{\sum_{t=1}^{N}n^{t}_{c,\neg s}+N\beta}. \tag{7}$$

The principles behind Eq. (7) is that when the targeted vertex is assigned to community c, the negative samples should be the vertices with less relevance to this community. The uncorrelated probability of a vertex is computed by $P_{neg}(v)$. Then, combined with Eq. (7), the reversed community-vertex distribution $\boldsymbol{\psi}_{neg}$ of negative samples for the targeted vertex v_t can be formulated as:

$$\boldsymbol{\psi}_{neg}|\{P_{neg}(v_i)\}_{i=1,i\neq t}^{N}\sim Dir(P_{neg}(v_1),\ldots,P_{neg}(v_N)). \tag{8}$$

Finally, we can draw negative samples as follows:

$$v|\boldsymbol{\psi}_{neg}\sim Multinomial(\boldsymbol{\psi}_{neg}). \tag{9}$$

Compared with the frequency-based fixed distribution $P_{NS}(v)$ of the NS method in Eq. (4), the reversed community-vertex distribution $\boldsymbol{\psi}_{neg}$ learned by our model can exploit the hierarchical structures of latent communities and draw negative samples with considering relevant vertex information, i.e., sampling vertices with uncorrelated probabilities toward the community assigned to the targeted vertex. In the following section, we will introduce how our proposed HNS can optimize the Skip-Gram objective.

3.5. Optimized objective analysis

To begin with, we follow Mikolov et al. [22] to use stochastic gradient descent (SGD) for optimization. Combined with Eq. (4), the objective of NS in Eq. (3) can be further derived as:

$$\begin{aligned}\mathcal{L}_{NS}&=\log(\sigma(\mathbf{v}'_j\cdot\mathbf{v}_i))+\sum_{k=1}^{K}\mathbb{E}_{v_k\sim P_{NS}(v)}\log(\sigma(-\mathbf{v}'_k\cdot\mathbf{v}_i))=\log(\sigma(\mathbf{v}'_j\cdot\mathbf{v}_i))+K\sum_{v_k\in V}P_{NS}(v_k)\log(\sigma(-\mathbf{v}'_k\cdot\mathbf{v}_i))\\&=\log(\sigma(\mathbf{v}'_j\cdot\mathbf{v}_i))+\sum_{v_k\in V}\frac{Kd_{v_k}^{\beta}}{\sum_{n\in V}d_n^{\beta}}\log(\sigma(-\mathbf{v}'_k\cdot\mathbf{v}_i))=-\log(1+e^{\mathbf{v}'_j\cdot\mathbf{v}_i})+\mathbf{v}'_j\cdot\mathbf{v}_i+\sum_{v_k\in V}\frac{-Kd_{v_k}^{\beta}}{\sum_{n\in V}d_n^{\beta}}\log(1+e^{\mathbf{v}'_k\cdot\mathbf{v}_i})\\&=-\sum_{v_k\in V}\left(\frac{Kd_{v_k}^{\beta}}{\sum_{n\in V}d_n^{\beta}}+\mathbf{1}_{v_k==v_j}\right)\log(1+e^{\mathbf{v}'_k\cdot\mathbf{v}_i})+\mathbf{v}'_j\cdot\mathbf{v}_i,\end{aligned} \tag{10}$$

where v_i is the targeted vertex, and v_j is its context vertex. In HNS, we adopt $\boldsymbol{\psi}_{neg}$ to draw negative samples by using P_{neg} in Eq. (7) to replace P_{NS} in Eq. (10). Similarly, we get the objective of HNS as:

$$\mathcal{L}_{HNS}=-\sum_{v_k\in V}(K\cdot P_{neg}(v_k)+\mathbf{1}_{v_k==v_j})\log(1+e^{\mathbf{v}'_k\cdot\mathbf{v}_i})+\mathbf{v}'_j\cdot\mathbf{v}_i. \tag{11}$$

To optimize the above objective, we can get its derivative as:

$$\frac{\partial\mathcal{L}_{HNS}}{\partial\mathbf{v_k}}=\begin{cases}[1-(K\cdot P_{neg}(v_j)+1)\frac{1}{1+e^{-\mathbf{v}'_j\cdot\mathbf{v}_i}}]\mathbf{v_i}, & \text{if } v_k==v_j\\ -K\cdot P_{neg}(v_k)\frac{1}{1+e^{-\mathbf{v}'_k\cdot\mathbf{v}_i}}\mathbf{v_i}, & \text{if } v_k\neq v_j\end{cases} \tag{12}$$

Here, we want to explain how our HNS optimizes the objective. Let the derivative $\frac{\partial \mathcal{L}_{HNS}}{\partial \mathbf{v_k}} = 0$ or $\rightarrow 0$, we can get the following expressions:

$$\begin{cases} \mathbf{v}_j' \cdot \mathbf{v}_i = -\log(K \cdot P_{neg}(v_j)), & \text{if } v_k == v_j \text{ and } \frac{\partial \mathcal{L}_{HNS}}{\partial \mathbf{v_k}} = 0 \\ 1 - P_{neg}(v_k) \rightarrow 0, & \text{if } v_k \neq v_j \text{ and } \frac{\partial \mathcal{L}_{HNS}}{\partial \mathbf{v_k}} \rightarrow 0 \end{cases} \tag{13}$$

where we can see that: (1) For each targeted vertex v_i, we optimize the similarity between it and its positive pairs, e.g., v_j in Eq. (13), to approximate $-\log(K \cdot P_{neg}(v_j))$ which is proportional to $P(v_j|z_{s,v_i} = c, \cdot)$ represented the probability of vertex v_j belonging to the community c assigned to the targeted vertex v_i; (2) For v_i and its negative samples, e.g., v_k, we intend to optimize the similarity between them to approximate $1 - P_{neg}(v_k) \rightarrow 0$, which indicates that negative vertex v_k is less relevant to the community c assigned to the targeted vertex v_i. The reversed community-vertex distribution ψ_{neg}, as mentioned in Eq. (8), is conducive to draw negative vertices with the probabilities $P_{neg}(v_k)$ close to 1, i.e., $1 - P_{neg}(v_k)$ is close to 0. To sum up, from Eq. (13), we can observe that our proposed HNS can help to optimize the objective of NS by incorporating the latent community structures of relevant vertex information. The detailed pseudocode is shown in Algorithm 1.

Algorithm 1: Training Process of HNS

Input: graph $G = (V, E)$, iteration number I_1 and I_2, grahical model parameters α, β, γ, window size t, negative sample size K
Result: vertex embedding $\mathbf{v}$, community size C

```
begin
    def Gibbs_sampling(v):
        Draw a community assignment z_{s,v} for vertex v with Eq. 5 and
         Eq. 6;
        if z_{s,v} == C + 1 then
            C = C + 1;
        end
        Update the parameters n_s^{z_{s,v}} and n_{z_{s,v}}^v;
        return z_{s,v};
    end
    //Initialization;
    C = 1;
    Obtain walk seqeunces S from random walks;
    Initialize vector v for each vertex;
    Assign all vertices in S to the same community c = 1;
    for iter = 1 : I_1 do
        // Follow the generative process mentioned in 3.2;
        for each vertex v_i in each sequence s ∈ S do
            Gibbs_sampling(v_i);
        end
    end
    //Vertex embeddings;
    for iter = 1 : I_2 do
        for each vertex v_i in each sequence s ∈ S do
            Get its community assignment c = Gibbs_sampling(v_i);
            Draw the negative samples {v_k}_{k=1}^K from community c by
             calculating Eq. 9 with Eq. 7 and Eq. 8;
            for each j ∈ [i − t : i + t] do
                Using SGD to optimize the objective L_HNS with Eq. 12
            end
        end
    end
end
```

3.6. Complexity analysis

In HNS, the training process consists of two parts, including Gibbs Sampling [14] of the graphical model inference and vertex representation learning. The complexity of the first part is $O(I_1CVSL)$, where I_1 is the iteration number, C denotes the community size, V represents the vertex size, S is the number of sequences, and L is the length of the sequences. Then, the complexity of the second part is $O(I_2V\log V)$, where I_2 is the iteration number. Therefore, the total complexity of HNS is $O(V(I_1CSL+I_2\log V))$. Moreover, in practice, each part of HNS can be accelerated by the existing algorithms, e.g., PLDA [42], PLDA+ [19], and Skip-Gram [21]. These parallel algorithms only take linear time to train in large-scale scenarios. We leave these implements in our future work.

4. Experiment

In our experiments, we evaluate the performance of our proposed model in terms of vertex classification on three real-world networks. Moreover, we also investigate the robustness of the performance of HNS and its variant to their parameters.

4.1. Datasets

We conduct experiments on two citation networks and a language network. The detailed statistics are listed in Table 1.

(1) Cora[2] is a typical citation network constructed by [20]. After filtering out papers without labels, this dataset contains 2211 abstracts of machine learning papers from 7 classes and 5214 links among them. These links are citation relationships between papers.

(2) DBLP[3] consists of bibliography data in computer science constructed by [30]. DBLP is a citation network since each of its papers may cite or be cited by other papers. In our experiments, we select a list of conferences from 4 research fields: 1) database including SIGMOD, ICDE, VLDB, EDBT, PODS, ICDT, DASFAA, SSDBM, CIKM; 2) data mining including KDD, ICDM, SDM, PKDD, PAKDD; 3) AI including IJCAI, AAAI, NIPS, ICML, ECML, ACML, IJCNN, UAI, ECAI, COLT, ACL, KR; 4) computer vision including CVPR, ICCV, ECCV, ACCV, MM, ICPR, ICIP, ICME. After preprocessing, the retained dataset contains 17725 titles of papers from the above research areas and 52914 links.

(3) Wiki[4] is a language network that contains 2405 web pages from 17 categories and 17981 links between them. This dataset is firstly published from LBC[5] project and has been widely used for evaluating vertex classification tasks such as [44].

4.2. Models for comparison

To validate the performance of our model, we compare HNS with the state-of-the-art models and two variants.

NSCaching. NSCaching [49] is an improved generative adversarial network which can select high-quality negative samples by calculating their gradient magnitudes when using gradient descent to train vertex embeddings. The authors report that NSCaching can outperform state-of-the-art negative sampling methods.

NS. NS [22] is the most widely used model for vertex representation learning. It simplifies the objective of the Skip-Gram model but retains the quality, which works well for handling large-scale networks.

NN-NS. Non-neighbor Negative Sampling (NN-NS) is a variant of NS [22]. For each targeted vertex, NN-NS sets the probabilities of choosing its neighbors (corresponding to rank 1 neighbors mentioned in the introduction) as negative samples to zero. This is reasonable because it is unlikely that they are neighbors but not relevant.

HNS. HNS is the Hierarchical Negative Sampling model proposed in this paper, which is conducive to prevent drawing relevant vertices as the negative samples for the targeted vertex by modeling its neighbor information. The detailed description has been presented in Section 3.

HNS (w/o DP). HNS (w/o DP) is a variant of our proposed HNS without Dirichlet Process by fixing the community number as predefined. This variant is adopted as a reference line for HNS by manually searching for the optimal value of the number of communities. We try to evaluate whether HNS can automatically approximate this optimal value and achieve the comparable performance of HNS (w/o DP) in vertex classification tasks.

4.3. Parameter settings and implementation details

For a fair comparison, we uniformly set parameters in all methods as follows: walks per vertex are 80, window size is 10, the number of negative samples for each targeted vertex is 5, iteration number is 5, and the representation dimension is 200, which are the default settings recommended in [22,25]. In NSCaching, we select its ComplEx mode which can achieve the best performance. In HNS, we use grid search to select the following hyper-parameters:

[2] https://people.cs.umass.edu/∼mccallum/data.html.
[3] http://arnetminer.org/citation (V4 version is used).
[4] https://github.com/albertyang33/TADW.
[5] https://linqs.soe.ucsc.edu/.

Table 1
Statistics of the real-world networks.

Datasets	Cora	DBLP	Wiki
# Vertices	2211	17725	2405
# Edges	5214	52914	17981
# Labels	7	4	17

$\gamma \in \{0.5, 1.0, 1.5\}, \alpha \in \{0.02, 0.06, 0.1, 0.14, 0.18, 0.2\} \cdot (1e^{-2} \cdot \mathrm{M})$, and $\beta \in \{0.01, 0.05, 0.1\}$, where M is the total number of walk sequences. In the following section, we present the results of HNS with $(\gamma = 1.0, \alpha = 0.001 \cdot \mathrm{M}, \beta = 0.01)$. In HNS (w/o DP), we test the number of communities $C \in [10, 100]$ to search the optimal value. Note that we implement our work with Python language and run on a Linux server with Intel Core i9-9900 3.10 GHz CPU and 16 GM memory.

4.4. Evaluation metrics

In order to evaluate the overall classification performance across all the vertex classes, we adopt the Micro-F1 [28] as the evaluation metrics, which is formulated as:

$$\hat{Precision} = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C}(TP_i + FP_i)}, \quad \hat{Recall} = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C}(TP_i + FN_i)} Micro - F1 = 2 * \frac{\hat{Precision} * \hat{Recall}}{\hat{Precision} + \hat{Recall}} \tag{14}$$

where C denotes the number of classes, TP, FP, and FN denote True Positive, False Positive and False Negative, respectively. The value of Micro-F1 ranges from zero to one, where a higher value means a better classification performance.

4.5. Evaluation on vertex classification

For vertex classification, we take representations of vertices as features and employ one-vs-rest logistic regression implemented by Liblinear [11] to train classifiers. Then, we evaluate the classification accuracies of various methods with different ratios. Note that we adopt the training ratios from 1% to 10% to accelerate the training speed of classifiers and evaluate the performance of HNS under sparse scenes, which are also the evaluation settings used in [25]. For each training ratio, we randomly select vertices as a training set and the remaining vertices as a test set. We repeat this process 10 times and report the average accuracy.

As shown in Tables 2, 3, and 4, we present the classification performances of different methods on the datasets of Cora, DBLP, and Wiki in terms of Micro-F1 metrics with different training ratios. The highest scores are highlighted in bold. From these tables, we have the following observations:

(1) Our proposed models, HNS and HNS (w/o DP), consistently outperform the other baselines on different datasets with diverse training ratios, only with some exceptions in Wiki when the training ratios are 2%, 3%, and 4%, which demonstrates the effectiveness of our models. Moreover, HNS can achieve the similar or better performance of HNS (w/o DP) where a range of community numbers is applied to searching the optimal value (the details of the parameter influence on vertex classification tasks are reported in Section 4.6).
(2) In these tasks, especially on the Wiki dataset, we can see that NN-NS achieves higher scores than NS and NSCaching models in most cases. This is because the NN-NS model can exclude the neighbors when choosing the negative samples for each targeted vertex. Nevertheless, NN-NS only considers the case of rank 1 neighbors which is inevitable to be downgraded when vertices from rank 2 or 3 even N neighbors are also relevant.
(3) With the consideration of hierarchical latent structures in vertices, HNS and its variant are able to learn more discriminative vertex representations that are suitable in classification tasks. Furthermore, even with small training data, our proposed models still outperform the baseline methods, which demonstrates that they can handle the sparse situation better.

4.6. Parameter sensitivity

In this part, we want to study the influence of the parameters in HNS (w/o DP) and HNS, e.g., the number of communities C and the hyper-parameter α.

Firstly, we try to investigate the influence of community number for HNS (w/o DP) on vertex classification tasks to verify whether HNS can automatically approximate the optimal value of the community number. Here, we test HNS (w/o DP) by varying the numbers of community $C \in (\{10, 20, 40, 60, 80\}, \{10, 20, 50, 80, 100\}, \{10, 20, 30, 40, 50, 60\})$ on the datasets of Cora, DBLP, and Wiki, respectively. As shown in Fig. 4, we get the following observations:

(1) The Micro-F1 performance of HNS (w/o DP) fluctuates with the numbers of communities, especially on DBLP and Wiki datasets. Specifically, in Fig. 4a. Fig. 4b, and c, the curves of HNS (w/o DP) fluctuate at 60.14%, 64.39% and 49.85% in terms

Table 2
Micro-F1 values of vertex classification on Cora.

% Label Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
NSCaching	20.65	30.60	44.90	48.05	50.79	51.61	53.33	52.92	53.85	55.58
NS	38.15	45.18	55.10	57.32	58.64	60.46	59.94	58.97	59.07	60.25
NN-NS	42.30	41.58	55.99	56.76	58.12	58.78	59.70	58.82	59.61	59.60
HNS(w/o DP)	**42.39**	45.64	61.40	61.33	**63.35**	**64.02**	**65.29**	**64.82**	**65.42**	**65.98**
HNS	40.48	**47.99**	**61.63**	**61.89**	62.78	63.78	64.32	63.34	63.34	63.97

The number in bold indicates the best performance.

Table 3
Micro-F1 values of vertex classification on DBLP.

% Label Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
NSCaching	56.96	60.55	61.89	62.66	63.19	63.95	64.55	64.67	64.95	65.43
NS	58.79	60.14	61.06	61.83	62.68	63.30	63.63	63.45	63.81	64.26
NN-NS	60.66	60.82	60.90	60.24	60.66	61.17	61.67	61.46	61.94	62.62
HNS(w/o DP)	**62.40**	**64.04**	**64.69**	**64.87**	**64.92**	**64.93**	**65.30**	**65.36**	**65.54**	**65.58**
HNS	61.80	63.51	64.30	64.64	64.70	64.73	65.02	65.22	65.38	65.34

The number in bold indicates the best performance.

Table 4
Micro-F1 values of vertex classification on Wiki.

% Label Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
NSCaching	32.21	37.25	37.98	40.15	45.47	47.41	48.95	50.02	49.84	50.35
NS	36.62	**41.11**	42.78	47.08	50.55	51.66	53.51	54.95	55.05	55.43
NN-NS	37.51	40.90	**45.44**	**48.64**	52.12	53.16	54.18	55.58	55.50	55.52
HNS(w/o DP)	38.89	40.73	44.31	48.25	**53.74**	**54.40**	**55.70**	**56.85**	**56.97**	**57.41**
HNS	**40.02**	40.98	44.32	48.03	52.17	53.96	54.76	56.21	55.87	56.35

The number in bold indicates the best performance.

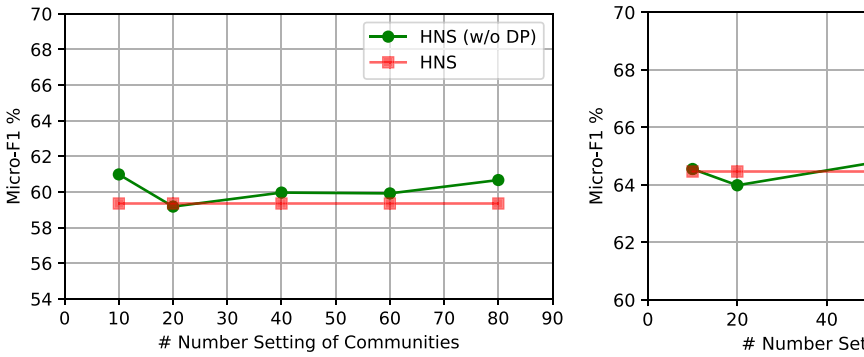
of Micro-F1 metrics, respectively. Besides, HNS (w/o DP) can achieve relatively stable and high performance with the numbers 40, 50, and 30 on Cora, DBLP, and Wiki accordingly.
(2) HNS can automatically approach the best performance of HNS (w/o DP). We can see that HNS achieves 59.35%, 64.46%, and 50.27% of Micro-F1 on three datasets, respectively. Moreover, note that the numbers of communities found by HNS are 39, 56, and 34, which verifies that HNS can approximate the optimal values of the community numbers set in HNS (w/o DP).

Secondly, we try to examine the influence of hyper-parameter α on the number of discovered communities and the Micro-F1 performances in HNS. We vary the values of $\alpha \in \{0.02, 0.06, 0.1, 0.14, 0.18, 0.2\} \cdot (1e^{-2} \cdot \mathrm{M})$ on three datasets, where M deNotes 2211, 17725 and 2405 in Cora, DBLP and Wiki, respectively. As shown in Fig. 5, we can obtain the followings:

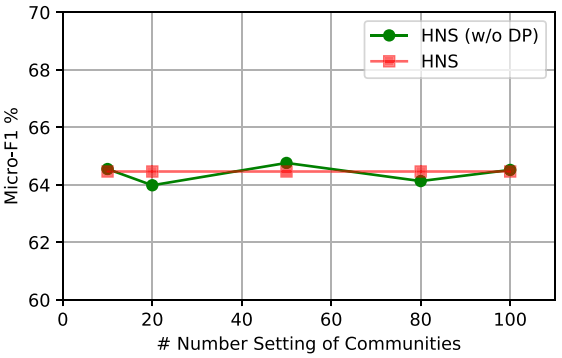
(1) Fig. 5a shows the numbers of communities found by HNS with different values of α. We can see that the discovered community number grows with α on three datasets. This is because α represents the probability of a vertex choosing a new community.
(2) Fig. 5b shows the Micro-F1 performances of HNS with different values of α. What needs to be mentioned is that, though the number of communities found by HNS increases along with α, its Micro-F1 performance on all datasets remains stable, which indicates that HNS is robust to the hyper-parameter α.

5. Related work

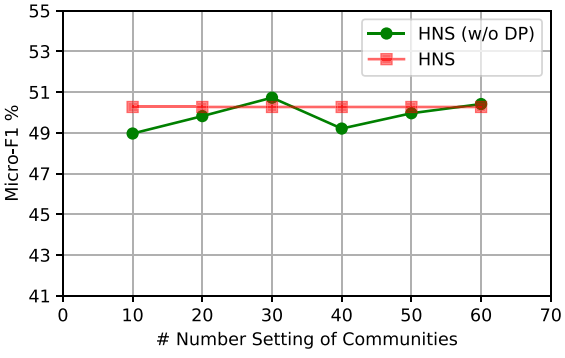
In this section, we introduce the related work on NRL and then present NS which is the most widely used optimization method in NRL. In addition, since we utilize the analogy between topics in texts and communities in networks [25,35] to model neighbor information of vertices, our work is also related to topic modeling and word embeddings. The analogy work will be introduced as well.



(a) Cora dataset

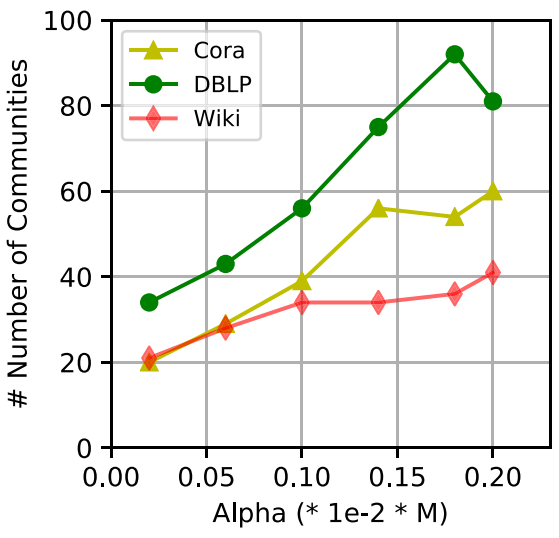


(b) DBLP dataset

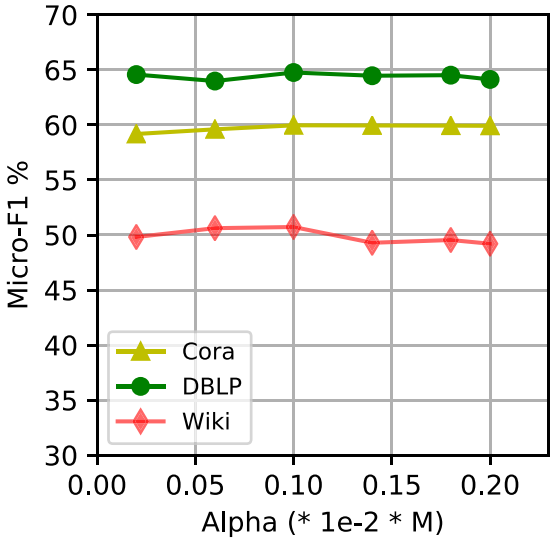


(c) Wiki dataset

Fig. 4. Vertex classification results of HNS (w/o DP) with different number settings of communities on three datasets. Each data point represents the average Micro-F1 performance of training ratios from 1% to 10% on the number setting of communities. Note that the performance of HNS remains stable since it can automatically infer the number of communities. The final discovered community numbers found by HNS on these datasets are 39, 56, and 34, respectively.



(a) Community Number with α



(b) Micro-F1 with α

Fig. 5. Influence of hyper-parameter α on the number of communities and the Micro-F1 performance obtained by HNS, respectively. Note that the number of M denotes the total number of walk sequences in each dataset, which is 2211, 17725 and 2405 in Cora, DBLP, and Wiki, respectively.

5.1. Network representation learning

With the rapid growth of social networks, a lot of work has been devoted to network analysis tasks [12]. Especially, NRL, i.e., network embeddings, shows its effectiveness in network analysis tasks, which generates many applications such as link prediction [15,34,10], vertex classification [25,44,48], and community detection [35,45].

In general, NRL aims to encode the structure information of vertices into a low-dimensional representation space and take the representations as features in further evaluation tasks. Firstly proposed by Perozzi et al. [25], DeepWalk performs random walks to explore the structural information in a network. The generated random walk sequences can be utilized to learn network embeddings with Skip-Gram model [21]. Then, a lot of work has been devoted to NRL [8]. For example, Tang et al. propose LINE [29] which employs a breadth-first search (BFS) strategy to discover vertex proximity. After that, Node2vec [15] is proposed to take both BFS and depth-first search (DFS) strategies into consideration and designs a biased random walk procedure to exploit broader neighborhoods. Moreover, Anton et al. [33] propose a versatile graph embedding that can explicitly calibrate to preserve vertex-to-vertex similarities in the embedding space. Though many attempts such as [25,29,15,36,50] are proposed to explore network topology and optimize objective functions for different purposes, they mostly rely on **Negative Sampling (NS)** [22] to approximate a variant of Skip-Gram objective because of the computational complexity. Notice that some alternatives such as Hierarchical Softmax [24] and Noise Contrastive Estimation [17,23] are proposed, but NS performs better in most cases [22,29,15,26]. However, all these alternative methods and NS draw negative samples from fixed distributions. The details of NS and the improved methods of NS-based NRL will be introduced in the following.

5.2. Negative sampling on NRL

NS is firstly mentioned in Skip-Gram model [21] which is the most widely used neural network language model for learning word embeddings. Since the Skip-Gram model requires a summation over the learned vectors of all vertices in each update, its objective is computationally infeasible in large-scale networks. NS is proposed to simplify the objective of Skip-Gram but retains its quality by drawing informative negative samples from a uniform distribution P_{NS} (more details are mentioned in Section 3.1). Up to now, NS has been applied on many applications including graph embeddings [8] and knowledge graph embeddings [38]. However, P_{NS} is a fixed distribution based on a popularity sampling strategy that vertices with higher frequencies are more likely to be drawn as the negative samples for each targeted vertex. This phenomenon may lead to low-quality embeddings especially for high degree vertices because a vertex has a larger probability to choose its neighbors with high degrees as its negative samples.

Then, to generate high-quality negative samples, some improved NS-based NRL methods are proposed. For instance, in R-NS [1], the authors define a sampler that a vertex cannot choose its neighbors as negative samples. Nevertheless, highly relevant vertices in rank N neighbors still can be drawn (an illustration is given in the introduction section). There are also some other methods to gain proper negative samples by selecting vertices with large gradient magnitudes when using gradient descent for optimization, e.g., [37,3,13,49] apply Generative Adversarial Networks (GAN) to choose negative samples in reinforcement learning manners. However, these GAN-based methods have high variance because they allow a vertex to choose its neighbor as a negative sample and the reinforcement-based techniques will induce error propagation and lead to unstable performance [43]. In general, all of the mentioned methods neither model the neighbor information of vertices nor select irrelevant vertices as negative samples for a targeted vertex.

5.3. Analogy on language models

After conducting random walks on a network, we can acquire a collection of walk sequences which share similar properties comparing with documents in a corpus [25,35]. Therefore, we can exploit the variants of topic models [2,32,16,5,47] for modeling vertex proximity information. Some previous work [35,18,27] explore latent topic/community structures as the self-embedded features into the learning space. However, these methods are not designed to model the neighbor information for negative sample selection. Moreover, they use a predefined topic/community number in the modeling, which is not appropriate in real world where the number is usually unknown. Note that in language models, there is also some previous work such as [7] which selects negative samples by ranking their gradient magnitudes to improve NS optimization. But these methods still allow a word to choose its neighbors as negative samples.

6. Conclusion and future work

In this paper, we propose a hierarchical negative sampling (HNS) method which can incorporate a graphical model to exploit the rank N neighbor information in negative sample selection. The learned distributions of reversed community-vertex can be regarded as the probabilities of vertices be drawn as negative samples for each targeted vertex, which can avoid setting rank numbers manually to define neighbor vertices. Moreover, we exploit Hierarchical Dirichlet Processes to explore the latent community structures of networks, which provides a nonparametric method for inferring the number

of communities. Experimental results show that our HNS can outperform the state-of-the-art models in terms of vertex classification tasks.

In future work, two main directions can be considered. First, we intend to exploit online topic modeling methods such as [5,6] with dynamic network embedding approaches [9] to generate desired negative samples in a dynamic network. Second, we would adopt our HNS to improve the representation learning performance of other machine learning applications [40,39] that inherently contain network structures.

CRediT authorship contribution statement

Junyang Chen: Conceptualization, Methodology, Software, Writing - original draft. **Zhiguo Gong:** Supervision, Writing - review & editing. **Wei Wang:** Visualization, Investigation. **Weiwen Liu:** Data curation, Writing - original draft.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by MOST (2019YFB1600704), FDCT (SKL-IOTSC-2018–2020, FDCT/0045/2019/A1), GSTIC (EF005/FST-GZG/2019/GSTIC), University of Macau (MYRG2017-00212-FST, MYRG2018-00129-FST).

References

[1] M. Armandpour, P. Ding, J. Huang, X. Hu, Robust negative sampling for network embedding, Proceedings of the AAAI Conference on Artificial Intelligence. 33 (2019) 3191–3198.
[2] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent dirichlet allocation, Journal of Machine Learning Research 3 (Jan) (2003) 993–1022.
[3] L. Cai, W.Y. Wang, Kbgan: Adversarial learning for knowledge graph embeddings, 2017. arXiv preprint arXiv:1711.04071.
[4] S. Cavallari, V.W. Zheng, H. Cai, K.C.-C. Chang, E. Cambria, Learning community embedding with community detection and node embedding on graphs, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 377–386.
[5] J. Chen, Z. Gong, W. Liu, A nonparametric model for online topic discovery with word embeddings, Information Sciences 504 (2019) 32–47.
[6] J. Chen, Z. Gong, W. Liu, A dirichlet process biterm-based mixture model for short text stream clustering, Applied Intelligence (2020) 1–11.
[7] L. Chen, F. Yuan, J.M. Jose, W. Zhang, Improving negative sampling for word representation using self-embedded features, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2018, pp. 99–107.
[8] P. Cui, X. Wang, J. Pei, W. Zhu, A survey on network embedding, IEEE Transactions on Knowledge and Data Engineering 31 (5) (2018) 833–852.
[9] L. Du, Y. Wang, G. Song, Z. Lu, J. Wang, Dynamic network embedding: An extended approach for skip-gram based network embedding, in: IJCAI, 2018. pp. 2086–2092.
[10] D.M. Dunlavy, T.G. Kolda, E. Acar, Temporal link prediction using matrix and tensor factorizations, ACM Transactions on Knowledge Discovery from Data (TKDD) 5 (2) (2011) 1–27.
[11] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, C.-J. Lin, Liblinear: A library for large linear classification, Journal of Machine Learning Research 9 (Aug) (2008) 1871–1874.
[12] A. Fischer, J.F. Botero, M.T. Beck, H. De Meer, X. Hesselbach, Virtual network embedding: A survey, IEEE Communications Surveys & Tutorials 15 (4) (2013) 1888–1906.
[13] H. Gao, H. Huang, Self-paced network embedding, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1406–1415.
[14] T.L. Griffiths, M. Steyvers, Finding scientific topics, Proceedings of the National academy of Sciences 101 (suppl 1) (2004) 5228–5235.
[15] A. Grover, J. Leskovec, node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855–864.
[16] J. Guo, Z. Gong, A nonparametric model for event discovery in the geospatial-temporal space, in: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, 2016, pp. 499–508.
[17] M.U. Gutmann, A. Hyvärinen, Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics, Journal of Machine Learning Research 13 (Feb) (2012) 307–361.
[18] Y. Liu, Z. Liu, T.-S. Chua, M. Sun, Topical word embeddings, in: Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.
[19] Z. Liu, Y. Zhang, E.Y. Chang, M. Sun, Plda+ parallel latent dirichlet allocation with data placement and pipeline processing, ACM Transactions on Intelligent Systems and Technology (TIST) 2 (3) (2011) 1–18.
[20] A.K. McCallum, K. Nigam, J. Rennie, K. Seymore, Automating the construction of internet portals with machine learning, Information Retrieval 3 (2) (2000) 127–163.
[21] Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
[22] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in neural information processing systems, 2013, pp. 3111–3119.
[23] A. Mnih, Y.W. Teh, A fast and simple algorithm for training neural probabilistic language models, 2012. arXiv preprint arXiv:1206.6426.
[24] F. Morin, Y. Bengio, Hierarchical probabilistic neural network language model, Aistats 5 (2005) 246–252, Citeseer.
[25] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 701–710.
[26] L.F. Ribeiro, P.H. Saverese, D.R. Figueiredo, struc2vec: Learning node representations from structural identity, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 385–394.
[27] B. Shi, W. Lam, S. Jameel, S. Schockaert, K.P. Lai, Jointly learning word embeddings and latent topics, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017, pp. 375–384.
[28] A. Sun, E.-P. Lim, Hierarchical text classification and evaluation, in: Proceedings 2001 IEEE International Conference on Data Mining. IEEE, 2001, pp. 521–528.
[29] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, Q. Mei, Line: Large-scale information network embedding, in: Proceedings of the 24th International Conference on World Wide Web, 2015, pp. 1067–1077.

[30] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, Z. Su, Arnetminer: extraction and mining of academic social networks, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, pp. 990–998.
[31] Y.W. Teh, Dirichlet process, Rncyclopedia of Machine Learning, 2011.
[32] Y.W.Teh, M.I. Jordan, M.J. Beal, D.M. Blei, Sharing clusters among related groups: Hierarchical dirichlet processes, in: Advances in Neural Information Processing Systems, 2005, pp. 1385–1392.
[33] A. Tsitsulin, D. Mottin, P. Karras, E. Müller, Verse: Versatile graph embeddings from similarity measures, in: Proceedings of the 2018 World Wide Web Conference, 2018, pp. 539–548.
[34] C. Tu, H. Liu, Z. Liu, M. Sun, Cane: Context-aware network embedding for relation modeling, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (vol. 1: Long Papers), 2017. pp. 1722–1731.
[35] C. Tu, X. Zeng, H. Wang, Z. Zhang, Z. Liu, M. Sun, B. Zhang, L. Lin, A unified framework for community detection and network representation learning, IEEE Transactions on Knowledge and Data Engineering 31 (6) (2018) 1051–1065.
[36] K. Tu, P. Cui, X. Wang, P.S. Yu, W. Zhu, Deep recursive network embedding with regular equivalence, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2357–2366.
[37] P. Wang, S. Li, R. Pan, Incorporating gan for negative sampling in knowledge representation learning, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[38] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: A survey of approaches and applications, IEEE Transactions on Knowledge and Data Engineering 29 (12) (2017) 2724–2743.
[39] W. Wang, J. Chen, J. Wang, J. Chen, Z. Gong, Geography-aware inductive matrix completion for personalized point of interest recommendation in smart cities, IEEE Internet of Things Journal. (2019).
[40] W. Wang, J. Chen, J. Wang, J. Chen, J. Liu, Z. Gong, Trust-enhanced collaborative filtering for personalized point of interests recommendation, IEEE Transactions on Industrial Informatics (2019).
[41] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, S. Yang, Community preserving network embedding, in: Thirty-first AAAI Conference on Artificial Intelligence, 2017.
[42] Y. Wang, H. Bai, M. Stanton, W.-Y. Chen, E.Y. Chang, Plda: Parallel latent dirichlet allocation for large-scale applications, in: International Conference on Algorithmic Applications in Management, Springer, 2009, pp. 301–314.
[43] R.J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine Learning 8 (3–4) (1992) 229–256.
[44] C. Yang, Z. Liu, D. Zhao, M. Sun, E. Chang, Network representation learning with rich text information, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.
[45] J. Yang, 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, 2013, pp. 587–596.
[46] J. Yang, J. McAuley, J. Leskovec, Community detection in networks with node attributes, in: 2013 IEEE 13th International Conference on Data Mining, IEEE, 2013, pp. 1151–1156.
[47] J. Yin, J. Wang, A model-based approach for text clustering with outlier detection, in: 2016 IEEE 32nd International Conference on Data Engineering (ICDE), IEEE, 2016, pp. 625–636.
[48] C. Zhang, A. Swami, N.V. Chawla, Shne: Representation learning for semantic-associated heterogeneous networks, in: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, 2019, pp. 690–698.
[49] Y. Zhang, Q. Yao, Y. Shao, L. Chen, Nscaching: Simple and efficient negative sampling for knowledge graph embedding, in: 2019 IEEE 35th International Conference on Data Engineering (ICDE), IEEE, 2019, pp. 614–625.
[50] Z. Zhang, P. Cui, X. Wang, J. Pei, X. Yao, W. Zhu, Arbitrary-order proximity preserved network embedding, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2778–2786.