Topic-aware latent models for representation learning on networks

Abdulkadir Çelikkanat, Fragkiskos D Malliaros

Topic-aware latent models for representation learning on networks

Abdulkadir Çelikkanat[a], Fragkiskos D. Malliaros[a]

[a]*Paris-Saclay University, CentraleSupélec, Inria, Centre for Visual Computing, 91192 Gif-Sur-Yvette, France*

Abstract

Network representation learning (NRL) methods have received significant attention over the last years thanks to their success in several graph analysis problems, including node classification, link prediction and clustering. Such methods aim to map each vertex of the network into a low dimensional space in a way that the structural information of the network is preserved. Of particular interest are methods based on random walks; such methods transform the network into a collection of node sequences, aiming to learn node representations by predicting the context of each node within the sequence. In this paper, we introduce TNE, a generic framework to enhance the embeddings of nodes acquired by means of random walk-based approaches with topic-based information. Similar to the concept of topical word embeddings in Natural Language Processing, the proposed model first assigns each node to a latent community with the favor of various statistical graph models and community detection methods, and then learns the enhanced topic-aware representations. We evaluate our methodology in two downstream tasks: node classification and link prediction. The experimental results demonstrate that by incorporating node and community embeddings, we are able to outperform widely-known baseline NRL models.

Keywords: Network representation learning, Node embeddings, Link prediction, Community structure

1. Introduction

Graphs are important mathematical structures commonly used to represent objects and their relations in real-world systems such as the World Wide Web, social networks, and biological networks. Of particular importance is how to deal with learning tasks on graphs, such as the ones of friendship recommendation in social networks and protein function prediction in protein-protein interaction networks—with the major challenge being how to incorporate information about the structure of the graph in the learning process. To this direction, a plethora of approaches have emerged under the area of *network representation learning* (NRL) [16]. The main goal of NRL models is to learn feature vectors corresponding to the nodes of the graph (also known as *node embeddings*), by preserving important structural properties of the network; those vectors can later be used to perform various analysis and mining tasks including visualization, node classification and link prediction with the favor of machine learning algorithms.

Initial studies in the field of network representation learning, have mostly relied on *matrix factorization* techniques, since various properties and interactions between nodes can be expressed as matrix operations. Nevertheless, such methods are not scalable to large-scale networks mainly due to their increased running time complexity—especially for graphs consisting of millions of nodes and edges [16]. More recent studies have concentrated on developing methods suitable for relatively large-scale networks – being able to effectively approximate the underlying objective functions that capture meaningful information about the nodes of the graph and their properties.

Email addresses: abdcelikkanat@gmail.com (Abdulkadir Çelikkanat), fragkiskos.malliaros@centralesupelec.fr (Fragkiskos D. Malliaros)

Accepted to Pattern Recognition Letters

Many node representation learning methods have been inspired by the advancements in the area of *natural language processing* (NLP), borrowing various ideas originally developed for computing *word embeddings*. A prominent example here is the *Skip-Gram* architecture [22], which aims to find latent representations of words by estimating their context within the sentences of a textual corpus. To this direction, many pioneer studies in NRL [26, 15] utilize the idea of random walks to transform graphs into a collection of sentences – as an analogy to the area of natural language – and these sentences or walks are later being used to learn node embeddings.

Although random walk-based approaches are strong enough to capture local connectivity patterns, they mainly suffer to sufficiently convey information about more global structural properties. More precisely, real-world networks have an inherent clustering (or community) structure, which can be utilized to further improve the predictive capabilities of node embeddings. One can interpret such structural information based on an analogy to the concept of *topics* in a collection of documents. In a similar way as word embeddings can be enhanced with topic-based information [21], here we aim at empowering node embeddings by employing information about the latent community structure of the network—that can be achieved by a process similar to the one of *topic modeling*.

In this paper, we propose *Topical Node Embeddings* (TNE), a framework in which node embeddings are enhanced with topic (or community) information towards learning topic-aware node representations—something that leads to further improvements in the performance on downstream tasks. Local clustering patterns are of great importance on many applications, enabling to better grasp hidden information of the network. For instance, consider two individuals sharing common friends or interests, that are not yet represented by a direct link in the network. A careful analysis of the local community structure can further help to infer the missing information (e.g., missing links), and therefore boost the predictive capabilities of node embeddings. Motivated by that, the proposed TNE framework aims to directly leverage the latent community structure of the graph while learning node embedding vectors. The main contributions of the paper can be summarized as follows:

- *Latent graph models and topic representations*. We show how existing latent space discovery models, such as community detection and topic models, can be incorporated in the node representation learning process.
- *Node representation learning framework*. We propose a new model, called TNE, which first learns community embeddings from the graph, and then uses them to improve the node representations extracted by random walk-based methods. We examine various instances of this model, studying their properties.
- *Enriched feature vectors*. We perform a detailed empirical evaluation of the embeddings learned by TNE on the tasks of node classification and link prediction. As the experimental results indicate, the proposed model learn feature vectors which can boost the performance on downstream tasks.

The rest of the paper is organized as follows. Section 2 describes the related work, and in Section 3, we give the fundamental concepts for unfamiliar readers and formulate the problem. In Section 4, we describe the concept of topical node representation learning. The proposed TNE model is presented in Section 5. Section 6 presents the experimental results, and finally, in Section 7 we conclude our work providing also future research directions.

2. Related work

Several methods have been proposed to learn latent node representations in an unsupervised manner [16, 5]. Traditional unsupervised feature learning methods typically aim at factorizing a matrix representation, chosen in a way to properly consider the underlying properties of a given network. Characteristic examples here constitute models that preserve first-order proximity of nodes, such as Laplacian Eigenmaps [1] and IsoMap [35]. More recently, algorithms including LINE [34], GraRep [6] and HOPE [24] were designed to preserve higher-order proximities of nodes. Nevertheless, despite the fact that matrix factorization approaches offer an elegant way to capture the desired properties, they mainly suffer from their time complexity.

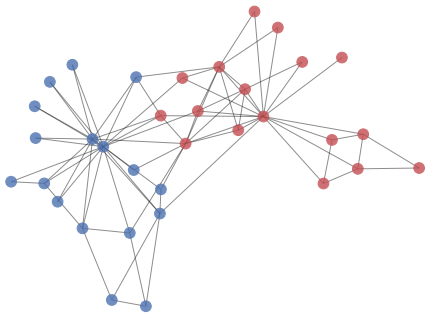


Figure 1: The topic-community assignments in Zachary's karate club network. Each node v is assigned to a community label z maximizing the posterior probability $Pr(z|v)$ by TNE-GLDA model.

Random walk-based methods [16], including NODE2VEC [15], DEEPWALK [26], have gained considerable attention, mainly due their elegant way to define proximity among nodes based on random walks, as well the computational efficiency of the *Skip-Gram* model. Following this line of research, various extensions have been proposed [20, 30, 10, 23, 12, 9, 8]. Lastly, it was recently shown that that DEEPWALK and NODE2VEC implicitly perform matrix factorizations [28, 29].

To the best of our knowledge, very few models benefit from the community structure of real networks while learning embeddings. The COME model [7] proposes a closed-loop procedure among the encoding of communities, learning node embeddings and community detection in the network. M-NMF [38] targets to learn node representations by incorporating community structure information in a non-negative matrix factorization formulation. COSINE [40] is a generative model learning the social network embeddings from information diffusion cascades. A recent approach, GEMSEC [32], learns node embeddings with an explicitly defined community preserving objective function. As we will present shortly, our work aims at independently learning node and community (topic) embeddings, and then combining them into expressive topical feature vectors.

3. Background concepts

We will use $G = (\mathcal{V}, \mathcal{E})$ to denote a graph, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ indicates the set of edges. Our goal is to find a mapping function $\beta : \mathcal{V} \rightarrow \mathbb{R}^{\mathcal{D}}$, where $\beta[v]$ will correspond to the representation of node v in a lower-dimensional space $\mathbb{R}^{\mathcal{D}}$; $\mathcal{D}$ is generally referred to as the embedding or dimension size, and is much smaller than the cardinality of the vertex set, $|\mathcal{V}|$.

Node embedding methods based on the popular *Skip-Gram* architecture (e.g., [15, 26, 23]) learn node representations using node sequences produced by random walks over a given network. They mainly target to maximize the likelihood of the occurrences of nodes within a certain distance with respect to each other, as follows:

$$\mathcal{F}_N(\alpha, \beta) := \arg\max_{\Omega} \sum_{\mathbf{w} \in \mathcal{W}} \sum_{1 \leq i \leq L} \sum_{-\gamma \leq j \neq 0 \leq \gamma} \log Pr(v_{i+j} \mid v_i; \Omega), \tag{1}$$

where $\mathcal{W}$ is the set of random walks $\mathbf{w} = (v_1, \ldots, v_i, \ldots, v_L) \in \mathcal{V}^L$, $\Omega = (\alpha, \beta)$ is the model parameters that we would like to learn and γ refers to the window size. Any nodes appearing inside the window size of *center* node v_i are referred to as *context* nodes. Note that, we obtain two different representation vectors $\alpha[v]$ and $\beta[v]$ for each node $v \in \mathcal{V}$, one per role of the node as center and context; in the experimental evaluation, we will only consider vector $\beta[v]$, which corresponds to the vector if the node is interpreted as a center node.

4. Learning topic representations

Complex networks, such as those arising from social or biological settings, consist of latent clusters of different sizes in which the nodes are more likely to be connected to each other [13, 19, 18]. Our main goal here is to use the latent clusters of a network in order to obtain enriched representations. This can be achieved by enhancing node embedding vectors with *topic representations*. By replacing a node v_i with its community label z_i in a random walk, we learn *community embeddings* by predicting the nodes in the context of a community label. More formally, we can define our objective function to learn topic representations as follows:

$$\mathcal{F}_T(\tilde{\alpha},\tilde{\beta}) := \arg\max_{\widetilde{\Omega}} \sum_{\mathbf{w}\in\mathcal{W}} \sum_{1\le i\le L} \sum_{-\gamma\le j\neq 0\le\gamma} \log Pr\left(v_{i+j} \mid z_i;\, \widetilde{\Omega}\right). \tag{2}$$

By maximizing the log-probability above, we obtain the embedding vectors corresponding to each community label $z_i \in \{1,\ldots,\mathcal{K}\}$, where $\mathcal{K}$ indicates the number of latent communities. In this work, we mainly use two approaches to detect latent communities. The first one is based on novel combination of generative statistical models accompanied with random walks, while the second one is based on traditional community detection algorithms that utilize the network structure itself.

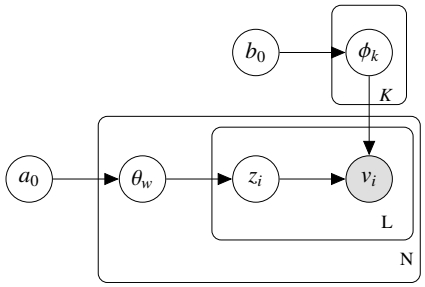
4.1. Random walks and generative graph models

Most real-world networks can be expressed as a combination of nested or overlapping communities [25]. Therefore, when a random walk is initialized, it does not only visit neighboring nodes but also traverses communities in the network (see Fig. 3b). In that regard, we assume that each random walk can be represented as random mixtures over latent communities, and each community can be characterized by a distribution over nodes. In other words, we can write the following generative model for each walk over the network:

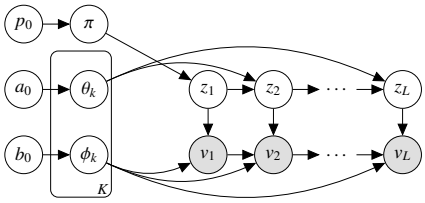
1. For each $k \in \{1,\ldots,\mathcal{K}\}$
   - $\phi_k \sim Dir(b_0)$
2. For each walk $\mathbf{w} = (v_1,\ldots,v_i,\ldots,v_L)$
   - $\theta_{\mathbf{w}} \sim Dir(a_0)$
   - For each vertex $v_i \in \boldsymbol{w}$
     - $z_i \sim Multinomial(\theta_{\mathbf{w}})$
     - $v_i \sim Multinomial(\phi_{z_i})$

Here, $\mathcal{N}$ is the number of walks and $\mathcal{L}$ is the walk length.

If we consider each random walk as a document and the collection of random walks as a corpus, it can be seen that the statistical process defined above corresponds to the well known Latent Dirichlet Allocation (LDA) model [3]. Therefore, each community corresponds to a distinct topic in the terminology of NLP (we use the terms *topic* and *community* interchangeably in the rest of the paper). We will refer to this model as GLDA (the plate representation is shown in Fig. 2a). As we show in Lemma 4.1, the relative frequency of the occurrences of a node in the generated walks is proportional to its degree in the network for large number of walks or walk lengths. This property was first empirically demonstrated in the work of [26], allowing *Skip-Gram* models to be applied on real-world graphs. Here we provide a formal argument of this empirical observation.



(a) GLDA model



(b) GHMM model

Figure 2: Plate representations of random walk-based topic representation models.

Lemma 4.1. *Let $G = (\mathcal{V},\mathcal{E})$ be a connected graph, and $\{X_l\}_{l\ge 1}$ be a Markov chain with state space $\mathcal{V} = \{1,\ldots,N\}$ and transition matrix P, where P_{ij} is defined as $1/d_i$ for each edge $(i,j) \in \mathcal{E}$ and 0 otherwise. If the Markov chain is aperiodic, then*

$$\lim_{L\to\infty} \frac{1}{L}\mathbb{E}\left[O_L^i\right] = \frac{d_i}{2|\mathcal{E}|},$$

where O_L^i is a random variable representing the number of occurrences of the node i in a random walk of length L.

Proof. Since the graph is connected, each state can be accessed by any other one. Thus, the Markov chain is also irreducible having a unique limiting distribution $\boldsymbol{\pi}$. Note that π_i is equal to $d_i/\sum_{k\in\mathcal{V}} d_k$ since it satisfies $\boldsymbol{\pi}P = \boldsymbol{\pi}$.

Then, we can write:

$$\lim_{L\to\infty}\frac{1}{L}\mathbb{E}\left[O_L^i\right] = \lim_{L\to\infty}\frac{1}{L}\mathbb{E}\left[\sum_{l=1}^{L}\mathbb{1}_{\{X_l=i\}}\right] = \lim_{L\to\infty}\frac{1}{L}\sum_{l=1}^{L}\mathbb{E}\left[\mathbb{1}_{\{X_l=i\}}\right]$$
$$= \lim_{L\to\infty}\frac{1}{L}\sum_{l=1}^{L} Pr\left(X_l = i\right) = \pi_i = \frac{d_i}{\sum_{k\in\mathcal{V}} d_k}$$
$$= \frac{d_i}{2|\mathcal{E}|},$$

where the equality in the second line follows from *Cesàro theorem* [11], since $Pr(X_l = i)$ converges to π_i as l goes to infinity. □

In the previous GLDA model, the latent community assignment of each node is independently chosen from the community label of the previous node in the random walk. However, the hidden state of the current node can play an important role towards determining the next vertex to visit, as the random walk also traverses through communities. Therefore, we can write the following generative process, by modifying the GLDA model:

1. For each $k \in \{1, \ldots, \mathcal{K}\}$
   - $\phi_k \sim Dir(b_0)$
   - $\theta_k \sim Dir(a_0)$
2. $\pi \sim Dir(p_0)$
3. For each walk $\mathbf{w} = (v_1, \ldots, v_i, \ldots, v_L)$
   - $z_1 \sim Dir(\pi)$
   - For each vertex $v_i \in \boldsymbol{w}$, for all $i < L$
     - $v_i \sim Multinomial(\phi_{z_i})$
     - $z_{i+1} \sim Multinomial(\theta_{z_i})$
   - $v_L \sim Multinomial(\phi_{z_L})$

The above model, in fact, corresponds to the well-known *Hidden Markov Model* (HMM) with symmetric Dirichlet priors over transition and emission distributions. In our experiments, we adopt the *Infinite Hidden Markov Model* (IHMM) [37] (we will refer to this model as GHMM; plate representation is shown in Fig. 2b). Note that, unlike the GLDA model, in the generation of each node sequence the same transition probabilities are used. Also, vectors θ_k and ϕ_k contain $\mathcal{K}$ and $|\mathcal{V}|$ components, respectively.

4.2. Network structure-based modeling

In the previous models, the generated random walks are used to detect the community (or topic) assignment of each node in the given node sequence. Here, we utilize two additional community detection models, which directly target to extract communities of nodes from a given network. The first one corresponds to the well-known LOUVAIN algorithm by [4] that extracts communities based on modularity maximization, while the second one to the BIGCLAM model for overlapping community detection [39].

5. Topical node embeddings

In this section, we will describe the proposed Topical Node Embeddings (TNE) model for learning topic-aware node representations. An overview of the model is given in Fig. 3. TNE aims to enhance node embeddings using information about the underlying topics of the graph obtained by the models described in Section 4. This can be achieved by learning node and topic embedding vectors independently of each other, jointly maximizing the objectives defined in Eq. (1) and (2). Combining these two objectives, we derive the following:

$$\max_{\Omega,\widetilde{\Omega}} \sum_{\mathbf{w}\in\mathcal{W}}\sum_{v_i\in\mathbf{w}}\sum_{-\gamma\le j\neq 0\le\gamma} \left[\log Pr\left(v_{i+j}|v_i;\Omega\right) + \log Pr\left(v_{i+j}|z_i;\widetilde{\Omega}\right)\right].$$

Skip-Gram models the probability measure in the above equation using the *softmax* function:

$$Pr\left(v_{i+j}|v_i\right) := \frac{\exp\left(\alpha[v_{i+j}]^\top\cdot\beta[v_i]\right)}{\sum_{u\in V}\exp\left(\alpha[u]^\top\cdot\beta[v_i]\right)}.$$

In our approach, though, we use the *sigmoid* function by adopting the negative sampling strategy [22], in order to make our computations more efficient. After obtaining the node and topic representations, our final step is to efficiently incorporate these two feature vectors, $\beta[v]$ and $\tilde{\beta}[z]$ of node v and community label z respectively, so as to obtain the final topic-enhanced node embedding. For this purpose, we concatenate node embedding vector $\beta[v]$ with the expected topic vector with respect to the distribution $p(\cdot|v_i)$. Our strategy can be formulated as follows:

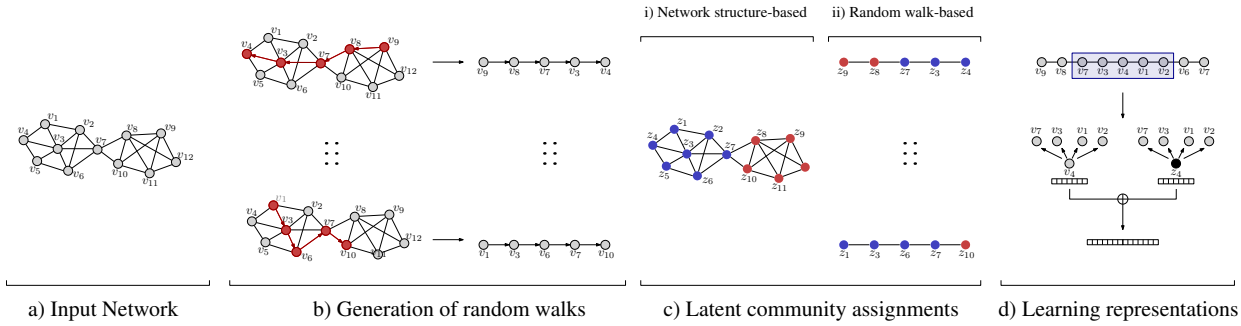


Figure 3: Schematic representation of the TNE model. The final representations are learned by combining node and topic embedding vectors. The representation of a node is learned using random walks performed over the network; its topic representation is similarly learned by assigning a topic/community label based either on random walks (TNE-GLDA, TNE-GHMM) or network structure-based approaches (TNE-LOUVAIN, TNE-BIGCLAM).

$$\beta[v_i] \oplus \sum_{k \in \{1,\ldots,\mathcal{K}\}} Pr(k|v) \cdot \tilde{\beta}[k], \tag{3}$$

where $\oplus$ indicates the concatenation operation. We refer to the final vector obtained after concatenating the node and topic feature vectors as *topical node embedding*. Algorithm 1 provides the pseudocode of the proposed TNE model.

Algorithm 1 Topical Node Embeddings (TNE)

Input: Graph $G = (\mathcal{V}, \mathcal{E})$; number of walks: $\mathcal{N}$; walk length: $\mathcal{L}$; window size: γ; number of communities: $\mathcal{K}$, topic representation learning method: T; node embedding size: $\mathcal{D}_n$; community embedding size: $\mathcal{D}_t$

Output: Embedding vectors of length $\mathcal{D}_n + \mathcal{D}_t$

1: Perform $\mathcal{N}$ random walks of length $\mathcal{L}$ for each node.
2: Learn node representations by optimizing Eq. (1).
3: Learn topic representations by optimizing Eq. (2), using any of the models T of Sec. 4.
4: Concatenate node and topic embeddings with Eq. (3).

The general structure of our framework follows. First, we need a collection of walks over the network to learn node and topic embeddings; here we have utilized biased random walks, similar to NODE2VEC. We further produce node-context pairs and use *Skip-Gram* to learn node embeddings following Eq. (1). Then, we choose a strategy (shown as T in Alg. 1) to learn topic representations. This step is quite flexible in the formulation of TNE. One approach is to generate topic assignments z_i of each node $v_i \in \mathcal{V}$ in the walk $w \in \mathcal{W}$, based on the random walk-based generative graph models GLDA and GHMM, defined in Sec 4.1. Alternatively, we can directly infer the latent clustering structure based on BIGCLAM and LOUVAIN models, as described in Sec. 4.2. Lastly, we combine node and topic embeddings using Eq. (3) to obtain the final topical node embedding vectors. Depending on the method used to learn topical representations, we will refer to the corresponding instances of TNE as TNE-GLDA, TNE-GHMM, TNE-LOUVAIN and TNE-BIGCLAM.

5.1. Running time complexity analysis

The time complexity of TNE varies depending on the algorithm used to detect latent topics and communities. The node and topic representations can be learned in $O(\mathcal{D} \cdot |V| \cdot \mathcal{S})$ time [36] using the *negative sampling* technique for a given node sequences and topic assignments, where $\mathcal{S}$ is the number of samples. If the LOUVAIN algorithm is chosen, the communities can be detected in $O(|V| \cdot \log^2 |V|)$ operations, while $O(|V| \cdot \mathcal{K})$ steps are required for BIGCLAM when the number of communities is high. The models that rely on random walks and generative models (GLDA, GHMM), can be run in $O(\mathcal{N} \cdot L \cdot \mathcal{K} \cdot I)$, where I is the number of iterations [27, 37].

6. Experimental evaluation

In this section, we firstly present the experimental setup used in our study, describing the baseline methods and datasets used in the evaluation. We also provide details about parameter settings for TNE and baselines. The performance of the proposed model is examined in two downstream tasks: node classification and link prediction. Our model has been implemented in Python and the source code can be found at: `https://abdcelikkanat.github.io/projects/TNE/`.

6.1. Baseline methods

We evaluate the performance of TNE against seven widely used baseline methods. (*i*) DeepWalk applies *Skip-Gram* [26] with uniform random walks. (*ii*) Node2Vec [15] extends DeepWalk following biased random walks. (*iii*) LINE [34] aims at learning representations relying on first-order and second-order node proximity. (*iv*) HOPE generates embedding vectors by capturing higher order information of the network; in our experiments we have used the *Katz* index. (*v*) NetMF [28] is a matrix factorization approach based on the pointwise mutual information of node co-occurrences. (*vi*) GEMSEC [32] is a random walk model that learns the community structure and the embeddings simultaneously. (*vii*) Finally, M-NMF [38] learns embeddings via a modularity-preserving matrix factorization scheme.

6.2. Datasets

We have used seven different networks in our experiments. *CiteSeer* [10] and *Cora* [33] are citation networks constructed using articles as nodes, with edges representing citations. *DBLP* is a co-authorship graph, where an edge exists between nodes if two authors have co-authored at least one paper. The labels used for classification represent the research areas. *AstroPh* and *HepTh* are both collaboration networks built from the papers submitted to the *ArXiv* repository for Physics related topics. *Facebook* [17] is a social network obtained from a survey conducted via a Facebook application. Lastly, *Gnutella* [31] is the peer-to-peer file sharing network. In all cases, we consider networks as undirected to ensure the consistency of the experiments. Table 1 provides more detailed information about the datasets.

Table 1: Statistics of networks. $|\mathcal{V}|$: number of nodes, $|\mathcal{E}|$: number of edges, $|\Lambda|$: number of labels and $|C|$: number of connected components.

	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\Lambda\rvert$	$\lvert C\rvert$	**Avg. Degree**	**Type**
CiteSeer	3,312	4,660	6	438	2.814	Citation
Cora	2,708	5,278	7	78	3.898	Citation
DBLP	27,199	66,832	4	2,115	4.914	Co-authorship
AstroPh	17,903	19,7031	-	1	22.010	Collaboration
HepTh	8,638	24,827	-	1	5.7483	Collaboration
Facebook	4,039	88,234	-	1	43.6910	Social
Gnutella	8,104	26,008	-	1	6.4186	Peer-to-peer

6.3. Parameter settings

In this section, we describe the parameters' settings that we have used for our experiments and clarify the strategies that we follow. To be consistent with the related literature, we have considered walks of length $\mathcal{L} = 10$, number of walks $\mathcal{N} = 80$ and window size $\gamma = 10$ for GEMSEC and for all random walk-based approaches. The remaining parameters of the baseline methods are set to their default values with the embedding size of 128. The difference instances of TNE are fed with biased random walks similar to those used in Node2Vec, setting hyper-parameters p, q to 1.0. To speed up the training process, we adopt the negative sampling [22] strategy. Stochastic Gradient Descent has been used for optimization, setting the initial learning rate to 0.0025 and its minimum value to 10^{-5}. We learn topic and node embedding vectors of sizes 32 and 96, respectively, so as the obtain feature vectors of length 128.

In the learning process of topic assignments of nodes with the TNE-Glda model, we perform collapsed Gibbs sampling [14] for parameter estimation and for inference. MCMC approach is used for the parameter estimation of TNE-Ghmm, with beam sampling for latent sequence resampling steps [37]. The number of topics for TNE-Glda is set to 100 for all networks except *Citeseer* and *Cora*, which is set to 150 and 125, respectively. The initial number of states for TNE-Ghmm is set to 20. For M-NMF and GEMSEC, we have performed parameter tuning for the number of communities over $\{5, 15, 20, 25, 50, 75, 100\}$.

We have performed all the experiments on an Intel Xeon 2.4GHz CPU server (32 Cores) with 60GB of memory. In order to examine the exact running time of the TNE variants, we have performed an experiment on artificially generated Erdös-Rényi random graphs of varying sizes, ranging from 2^8 to 2^{13} nodes. The running times

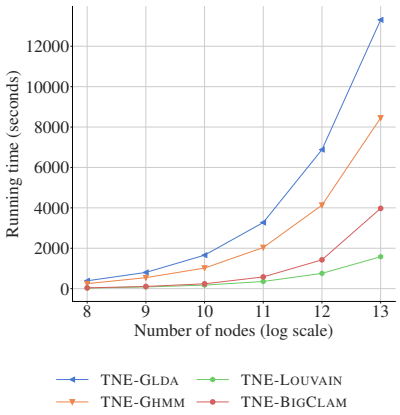
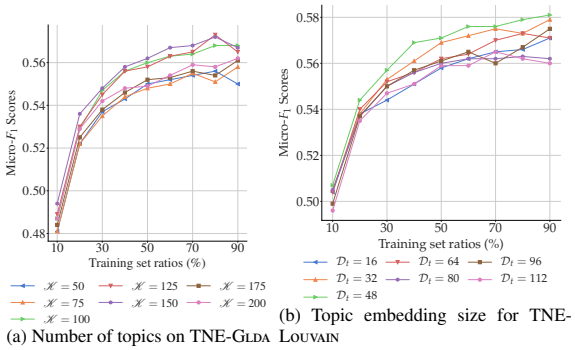


Figure 4: Running time of TNE on Erdös-Rényi random graphs of different size.



(a) Number of topics on TNE-GLDA

(b) Topic embedding size for TNE-LOUVAIN

Figure 5: Parameter sensitivity of TNE-GLDA and TNE-LOUVAIN models.

are reported in Figure 4. As expected, the TNE-LOUVAIN is the most scalable instance, due to the efficient way to infer the community structure.

6.4. Node classification

Experimental setup. In the node classification task, nodes are associated with labels and the goal is to predict the correct labels, observing only certain fraction of the network. We split the collection of feature vectors into training and tests sets, and apply an one-vs-rest logistic regression classifier with *L2* regularization for optimization. In order to provide more reliable experimental results, the same procedure is repeated for 50 times.

Results. The detailed results for the different instances of TNE framework as well as for the baseline methods are provided in Tables 2, 3 and 4. Experiments are reported for different ratios of training data (10%-90%). For each model, the first row corresponds to Micro-F_1 scores, while the second one to Macro-F_1 scores.

As we can observe, the TNE-LOUVAIN and TNE-BIGCLAM models perform quite well, outperforming most of the baseline methods for almost all different training ratios over *CiteSeer* and *DBLP* datasets. In the case of the *Cora* network, TNE-GLDA performs better especially when the train-test split of the dataset is quite balanced. The overall better performance of the two network structure-based instances TNE-LOUVAIN and TNE-BIGCLAM over random walk-based models (e.g., TNE-GLDA), can possibly be explained by the way that latent communities are extracted. While with TNE-GLDA and TNE-GHMM we are able to utilize random walks for both node and topic representations, it seems that those random walk are not expressive enough to recover the clustering structure as effectively as algorithms tailored to this task, such as BIGCLAM and LOUVAIN.

6.5. The effect of number of topics and topic embedding sizes

We have further analyzed the effect of the chosen number of topics $\mathcal{K}$ on the performance of the TNE-GLDA model on the *CiteSeer* network. As shown in Figure 5a, the increase in the number of topics makes positive contribution to the classification performance up to a certain level for TNE-GLDA model, reaching its highest F_1-score for $\mathcal{K} = 150$. The chosen number of topics seems to be a more crucial parameter especially for larger training data sizes.

The size of the topic embedding vector learned by Eq. (2) is another factor affecting the performance of TNE. In this paragraph, we examine the influence of the topic embedding size on the TNE-LOUVAIN model over *Citeseer* network. In the experiment, we have fixed the total embedding length to $\mathcal{D}_n + \mathcal{D}_t$ to 128, and we vary the length of topic ($\mathcal{D}_t$) and node ($\mathcal{D}_n$) embeddings. As it can be observed in Figure 5b, for small node embedding sizes, the scores diminish drastically since topic embeddings alone do not convey sufficient information to effectively represent the nodes On the contrary, when we accurately

balance the lengths of topic and node embeddings (for $\mathcal{D}_t = 48$ in this particular case), the contribution of topic-enhanced embeddings over purely node embeddings is evident.

Table 2: Node classification for varying training sizes over *CiteSeer*. For each method, the rows indicates the Micro-F_1 and Macro-F_1 scores, respectively.

	2%	4%	6%	8%	10%	30%	50%	70%	90%
DeepWalk	0.517	0.550	0.569	0.580	0.587	0.593	0.596	0.597	0.597
	0.476	0.506	0.524	0.534	0.540	0.544	0.547	0.547	0.546
Node2Vec	0.543	0.571	0.583	0.591	0.595	0.600	0.600	0.601	0.603
	0.497	0.525	0.535	0.542	0.546	0.549	0.549	0.549	0.550
LINE	0.418	0.459	0.476	0.487	0.494	0.500	0.505	0.507	0.512
	0.367	0.407	0.423	0.434	0.440	0.445	0.448	0.450	0.454
HOPE	0.216	0.235	0.253	0.265	0.276	0.288	0.299	0.304	0.316
	0.075	0.099	0.121	0.136	0.150	0.164	0.178	0.186	0.202
NetMF	0.538	0.567	0.579	0.586	0.590	0.592	0.594	0.599	0.601
	0.487	0.516	0.528	0.535	0.538	0.540	0.542	0.547	0.548
GEMSEC	0.491	0.517	0.532	0.538	0.545	0.551	0.552	0.556	0.558
	0.451	0.476	0.488	0.494	0.497	0.501	0.499	0.502	0.501
M-NMF	0.382	0.423	0.439	0.448	0.449	0.455	0.456	0.461	0.457
	0.306	0.353	0.370	0.380	0.381	0.389	0.390	0.394	0.390
TNE-Glda	0.535	0.578	0.593	0.605	0.610	0.616	0.616	0.621	0.618
	0.494	0.536	0.548	0.558	0.562	0.567	0.568	0.572	0.567
TNE-Ghmm	0.526	0.557	0.568	0.575	0.583	0.584	0.583	0.592	0.594
	0.481	0.512	0.521	0.527	0.534	0.535	0.533	0.542	0.543
TNE-Louvain	**0.551**	**0.589**	**0.604**	**0.614**	**0.619**	**0.624**	**0.627**	**0.632**	**0.638**
	0.506	**0.547**	**0.560**	**0.570**	**0.576**	**0.580**	**0.582**	**0.588**	**0.594**
TNE-BigClam	0.546	0.580	0.597	0.607	0.612	0.613	0.619	0.621	0.624
	0.502	0.536	0.549	0.557	0.562	0.562	0.566	0.569	0.568

Table 3: Node classification for varying training sizes over *Cora*. For each method, the rows indicates the Micro-F_1 and Macro-F_1 scores, respectively.

	2%	4%	6%	8%	10%	30%	50%	70%	90%
DeepWalk	0.747	0.782	0.799	0.808	0.815	0.821	0.825	0.827	0.832
	0.735	0.771	0.788	0.798	0.806	0.811	0.815	0.816	0.821
Node2Vec	**0.770**	**0.800**	0.816	0.824	0.831	0.835	0.839	0.842	0.845
	0.755	**0.788**	0.804	0.813	0.820	0.824	0.828	0.830	0.832
LINE	0.609	0.673	0.701	0.719	0.728	0.735	0.741	0.743	0.747
	0.578	0.659	0.691	0.710	0.720	0.727	0.733	0.736	0.738
HOPE	0.302	0.303	0.302	0.302	0.302	0.303	0.304	0.303	0.306
	0.066	0.067	0.067	0.067	0.067	0.067	0.068	0.070	0.074
NetMF	0.747	0.786	0.802	0.809	0.814	0.820	0.821	0.824	0.823
	0.735	0.776	0.791	0.799	0.804	0.810	0.811	0.813	0.812
GEMSEC	0.601	0.643	0.674	0.698	0.714	0.728	0.735	0.741	0.744
	0.562	0.611	0.646	0.672	0.689	0.704	0.713	0.719	0.722
M-NMF	0.656	0.700	0.717	0.725	0.732	0.736	0.736	0.744	0.742
	0.638	0.687	0.706	0.716	0.722	0.728	0.728	0.735	0.734
TNE-Glda	0.751	0.794	**0.817**	**0.831**	**0.840**	**0.846**	**0.850**	0.851	**0.856**
	0.738	0.782	**0.807**	**0.821**	**0.829**	**0.836**	**0.837**	0.838	0.841
TNE-Ghmm	0.762	0.791	0.809	0.816	0.824	0.827	0.828	0.832	0.834
	0.746	0.780	0.798	0.805	0.812	0.815	0.816	0.819	0.823
TNE-Louvain	0.765	0.797	0.815	0.826	0.835	0.843	0.846	**0.853**	0.850
	0.749	0.782	0.802	0.813	0.822	0.830	0.832	**0.839**	0.836
TNE-BigClam	0.754	0.792	0.809	0.820	0.828	0.836	0.837	0.847	0.851
	0.739	0.780	0.798	0.810	0.818	0.826	0.828	0.838	0.838

Table 5: Area Under Curve (AUC) scores for the link prediction task.

	Citeseer	*Cora*	*DBLP*	*AstroPh*	*HepTh*	*Facebook*	*Gnutella*
DeepWalk	0.770	0.739	0.919	0.911	0.843	0.980	0.559
Node2Vec	0.778	0.742	0.919	0.913	0.846	0.980	0.678
LINE	0.718	0.694	0.930	0.969	0.844	0.967	0.650
HOPE	0.744	0.712	0.873	0.931	0.836	0.975	0.636
NetMF	0.758	0.745	0.883	0.839	0.856	0.811	0.615
GEMSEC	0.748	0.733	0.920	0.816	0.823	0.661	0.473
M-NMF	0.751	0.714	0.891	0.948	0.861	0.977	**0.680**
TNE-Glda	**0.809**	0.775	0.958	0.977	0.903	**0.993**	0.629
TNE-Ghmm	0.793	**0.806**	**0.959**	**0.979**	**0.908**	**0.993**	0.663
TNE-Louvain	**0.809**	0.780	**0.959**	0.977	0.905	**0.993**	0.637
TNE-BigClam	0.792	0.767	0.958	0.977	0.904	**0.993**	0.631

Table 4: Node classification for varying training sizes over *DBLP*. For each method, the rows indicates the Micro-F_1 and Macro-F_1 scores, respectively.

	2%	4%	6%	8%	10%	30%	50%	70%	90%
DeepWalk	0.616	0.627	0.630	0.632	0.633	0.634	0.635	0.635	0.636
	0.551	0.560	0.563	0.564	0.565	0.566	0.566	0.567	0.567
Node2Vec	0.623	0.633	0.636	0.638	0.639	0.640	0.641	0.641	0.641
	0.561	0.568	0.571	0.573	0.574	0.574	0.575	0.574	0.574
LINE	0.594	0.603	0.606	0.608	0.610	0.610	0.611	0.611	0.611
	0.522	0.532	0.536	0.538	0.540	0.540	0.541	0.541	0.541
HOPE	0.379	0.379	0.380	0.381	0.383	0.385	0.387	0.388	0.391
	0.138	0.139	0.140	0.143	0.146	0.149	0.152	0.156	0.160
NetMF	0.605	0.613	0.617	0.619	0.620	0.620	0.623	0.623	0.623
	0.522	0.528	0.530	0.532	0.531	0.531	0.533	0.533	0.533
GEMSEC	0.597	0.607	0.611	0.613	0.614	0.615	0.615	0.616	0.615
	0.523	0.531	0.534	0.534	0.535	0.535	0.537	0.537	0.536
M-NMF	0.551	0.563	0.568	0.571	0.574	0.574	0.577	0.577	0.579
	0.418	0.436	0.443	0.447	0.450	0.451	0.454	0.455	0.455
TNE-Glda	0.617	0.627	0.631	0.633	0.634	0.635	0.635	0.636	0.638
	0.554	0.560	0.563	0.565	0.565	0.566	0.566	0.567	0.569
TNE-Ghmm	0.619	0.629	0.632	0.634	0.635	0.636	0.636	0.636	0.634
	0.552	0.561	0.563	0.565	0.565	0.566	0.567	0.567	0.564
TNE-Louvain	**0.625**	**0.636**	**0.638**	**0.641**	**0.642**	0.641	0.642	0.643	**0.642**
	0.564	0.572	0.574	0.576	0.577	0.577	0.578	0.579	0.576
TNE-BigClam	**0.625**	0.635	**0.638**	**0.641**	0.641	**0.643**	**0.643**	**0.644**	**0.642**
	0.566	**0.573**	**0.576**	**0.579**	**0.578**	**0.580**	**0.579**	**0.581**	**0.579**

6.6. Link Prediction

Experimental setup. In the link prediction task, we have limited access to the edges of the network, and our goal is to predict the missing (unseen) edges between nodes. We divide the edge set of a given network into two parts to form training and test sets, by randomly removing 50% of the edges (the network remains connected during the process). The removed edges are later used as positive samples in the test set. The same number of node pairs which does not exist in the initial network is chosen to obtain negative samples for each training and test sets. The node embedding vectors are converted into edge features by using the *Hadamard* product. We perform experiments using the logistic regression classifier with *L2* regularization.

Results. Table 5 shows the area under curve (AUC) scores for link prediction. The different instances of TNE are able to outperform the baselines except *Gnutella*, and the TNE-GHMM model shows the best performance across multiple datasets.

6.7. *Further Empirical Analysis of* TNE

Here, we further analyze the behaviour of the various TNE instances on artificially generated networks, focusing on controlled classification experiments. We generate random graphs by following a similar approach as in [2]. In particular, we use the *stochastic block model* to construct graphs that consist of three clusters $\mathcal{A}$, $\mathcal{B}$ and C, each one of size $1,000$ nodes. Figure 6 provides a schematic representation of the process. An intra-community link is added with probability p, while an edge between communities $\mathcal{A}$ and $\mathcal{B}$ is added with probability q. We use an additional parameter c in order to introduce asymmetry for inter-community links for the community pairs $\mathcal{A}-C$ and $\mathcal{B}-C$. We have constructed three networks $(\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3)$ for parameters $p = [0.06, 0.065, 0.07]$, $q = [0.04, 0.035, 0.003]$ and $c = [1.25, 1.429, 1.667]$, respectively. In our classification experiments, node labels correspond to community assignments.

Table 6 provides the Micro-F_1 scores for different walk lengths on $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$, with 40% of the datasets used as training set. We perform uniform random walks following DEEPWALK's strategy, and the instances of TNE are fed with the same node sequences. As we can observe, the performance of the models increases depending on the walk length, since by increasing the number of center-context node pairs improves the ability of the models to capture the structural properties of the graph. Comparing the extreme cases of $\mathcal{G}_1$ and $\mathcal{G}_3$, the latter shows more distinguishable community structure. Therefore, TNE has better performance on $\mathcal{G}_3$ since the community detection algorithms are able to correctly assign topic-community labels. Among them, TNE-LOUVAIN is the best performing instance. It detects good modularity communities from the graph structure itself, and thus, contrary to TNE-GLDA and TNE-GHMM, it does not rely on the the generated random walk sequences. Furthermore, it is more successful to find the clusters of the stochastic block model graphs compared to TNE-BIGCLAM, since BIGCLAM focuses on overlapping communities that do not appear on these artificially generated graphs.

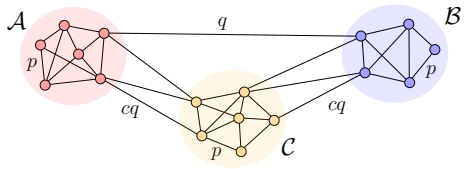


Figure 6: Illustration of link sampling in the stochastic block model.

Table 6: Micro-F_1 classification accuracy on the stochastic block model, with 40% training set ratio.

	$\mathcal{G}_1$			$\mathcal{G}_2$			$\mathcal{G}_3$		
Walk length, $\mathcal{L}$	10	50	100	10	50	100	10	50	100
DEEPWALK	0.708	0.721	0.720	0.859	0.856	0.861	0.914	0.931	0.936
TNE-GLDA	0.708	0.750	0.759	0.860	0.875	0.876	0.913	0.938	0.944
TNE-GHMM	0.707	0.752	0.762	0.861	0.877	0.875	0.916	0.938	0.944
TNE-LOUVAIN	**0.717**	**0.769**	**0.771**	**0.864**	**0.882**	**0.877**	**0.964**	**0.964**	**0.969**
TNE-BIGCLAM	0.710	0.749	0.757	0.856	0.873	0.872	0.912	0.933	0.940

7. Conclusions and future work

In this paper, we have proposed TNE, a topic-aware family of models for NRL. TNE takes advantage of the latent community structure of graphs, leading to the concept of topical node embeddings. We have examined four instances of the proposed model, that either use random walks to learn topic representations or rely on well-known community detection algorithms. TNE is capable of producing enriched representations, compared to traditional random walk-based approaches or matrix factorization-based models, leading to improved performance results in the tasks of node classification and link prediction.

As future work, we are interested to extend TNE towards utilizing the hierarchical community structure of real-world graphs, learning hierarchical node embeddings. Furthermore, we plan to examine the robustness of the learning representations with respect to changes on the structure of the graph.

Acknowledgments. We thank the anonymous reviewers for the constructive comments.

References

[1] Belkin, M., Niyogi, P., 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering, in: NIPS, Cambridge, MA, USA. pp. 585–591.

[2] Berberidis, D., Giannakis, G.B., 2019. Node embedding with adaptive similarities for scalable learning over graphs, in: IEEE Transactions on Knowledge and Data Engineering, pp. 1307–1321.

[3] Blei, D.M., Ng, A.Y., Jordan, M.I., 2003. Latent Dirichlet allocation. Journal of Machine Learning Research. 3, 993–1022.

[4] Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E., 2008. Fast unfolding of communities in large networks. J. Stat. Mech. 2008, P10008.

[5] Cai, H., Zheng, V.W., Chang, K., 2018. A comprehensive survey of graph embedding: problems, techniques, and applications. IEEE Transactions on Knowledge & Data Engineering 30, 1616–1637.

[6] Cao, S., Lu, W., Xu, Q., 2015. GraRep: learning graph representations with global structural information, in: CIKM, pp. 891–900.

[7] Cavallari, S., et al., 2017. Learning community embedding with community detection and node embedding on graphs, in: CIKM, pp. 377–386.

[8] Çelikkanat, A., Malliaros, F.D., 2019. Kernel node embeddings, in: GlobalSIP, pp. 1–5.

[9] Çelikkanat, A., Malliaros, F.D., 2020. Exponential family graph embeddings, in: AAAI, pp. 3357–3364.

[10] Chen, H., Perozzi, B., Hu, Y., Skiena, S., 2018. HARP: hierarchical representation learning for networks, in: AAAI, pp. 2127–2134.

[11] DeVito, C., 2007. Harmonic analysis: a gentle introduction. Jones and Bartlett Publishers.

[12] Epasto, A., Perozzi, B., 2019. Is a single embedding enough? learning node representations that capture multiple social context, in: WWW, pp. 394–404.

[13] Girvan, M., Newman, M.E.J., 2002. Community structure in social and biological networks. PNAS 99, 7821–7826.

[14] Griffiths, T.L., Steyvers, M., 2004. Finding scientific topics. PNAS 101.

[15] Grover, A., Leskovec, J., 2016. Node2vec: scalable feature learning for networks, in: KDD, pp. 855–864.

[16] Hamilton, W.L., Ying, R., Leskovec, J., 2017. Representation learning on graphs: methods and applications. IEEE Data Eng. Bull. 40, 52–74.

[17] Leskovec, J., Mcauley, J.J., 2012. Learning to discover social circles in ego networks, in: NIPS. Curran Associates, Inc., pp. 539–547.

[18] Li, H., Bu, Z., Wang, J., Cao, J., 2020. Dynamical clustering in electronic commerce systems via optimization and leadership expansion. IEEE Transactions on Industrial Informatics 16, 5327–5334.

[19] Li, H.J., Wang, L., Zhang, Y., Perc, M., 2020. Optimization of identifiability for efficient community detection. New Journal of Physics 22, 063035.

[20] Li, J., Zhu, J., Zhang, B., 2016. Discriminative deep random walk for network classification, in: ACL, pp. 1004–1013.

[21] Liu, Y., et al., 2015. Topical word embeddings, in: AAAI, pp. 2418–2424.

[22] Mikolov, T., et al., 2013. Distributed representations of words and phrases and their compositionality, in: NIPS, pp. 3111–3119.

[23] Nguyen, D., Malliaros, F.D., 2018. BiasedWalk: biased sampling for representation learning on graphs, in: Big Data, pp. 4045–4053.

[24] Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W., 2016. Asymmetric transitivity preserving graph embedding, in: KDD, pp. 1105–1114.

[25] Palla, G., et al., 2005. Uncovering the overlapping community structure of complex networks in nature and society. Nature 435, 814.

[26] Perozzi, B., Al-Rfou, R., Skiena, S., 2014. Deepwalk: online learning of social representations, in: KDD, pp. 701–710.

[27] Porteous, I., Newman, D., Ihler, A., Asuncion, A., Smyth, P., Welling, M., 2008. Fast collapsed Gibbs sampling for latent Dirichlet allocation, in: KDD, p. 569–577.

[28] Qiu, J., et al., 2018. Network embedding as matrix factorization: unifying deepwalk, line, pte, and node2vec, in: WSDM, pp. 459–467.

[29] Qiu, J., et al., 2019. Netsmf: large-scale network embedding as sparse matrix factorization, in: WWW, pp. 1509–1520.

[30] Ribeiro, L.F., Saverese, P.H., Figueiredo, D.R., 2017. Struc2vec: learning node representations from structural identity, in: KDD, pp. 385–394.

[31] Ripeanu, M., Iamnitchi, A., Foster, I., 2002. Mapping the gnutella network. IEEE Internet Computing 6, 50–57.

[32] Rozemberczki, B., Davies, R., Sarkar, R., Sutton, C., 2019. GEMSEC: graph embedding with self clustering, in: ASONAM, pp. 65–72.

[33] Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T., 2008. Collective classification in network data. AI magazine .

[34] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q., 2015. Line: large-scale information network embedding, in: WWW, pp. 1067–1077.

[35] Tenenbaum, J.B., Silva, V.d., Langford, J.C., 2000. A global geometric framework for nonlinear dimensionality reduction. Science 290, 2319–2323.

[36] Tsitsulin, A., Mottin, D., Karras, P., Müller, E., 2018. VERSE: versatile graph embeddings from similarity measures, in: WWW, p. 539–548.

[37] Van Gael, J., et al., 2008. Beam sampling for the infinite hidden markov model, in: ICML, Association for Computing Machinery. p. 1088–1095.

[38] Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., Yang, S., 2017. Community preserving network embedding, in: AAAI, pp. 203–209.

[39] Yang, J., Leskovec, J., 2013. Overlapping community detection at scale: a nonnegative matrix factorization approach, in: WSDM, pp. 587–596.

[40] Zhang, Y., Lyu, T., Zhang, Y., 2018. COSINE: community-preserving social network embedding from information diffusion cascades, in: AAAI, pp. 620–2627.