TNE: A Latent Model for Representation Learning on Networks

Abdulkadir Çelikkanat
Université Paris-Saclay
CentraleSupélec, Inria Saclay
Gif-sur-Yvette, France
abdcelikkanat@gmail.com

Fragkiskos D. Malliaros
Université Paris-Saclay
CentraleSupélec, Inria Saclay
Gif-sur-Yvette, France
fragkiskos.me@gmail.com

Abstract

Network representation learning (NRL) methods aim to map each vertex into a low dimensional space by preserving both local and global structure of a given network. In recent years, various approaches based on random walks have been proposed to learn node embeddings – thanks to their success in several challenging problems. In this paper, we introduce a general framework to enhance node embeddings acquired by means of the random walk-based approaches. Similar to the notion of *topical word embeddings* in NLP, the proposed framework assigns each vertex to a topic with the favor of various statistical models and community detection methods, and then generates the enhanced community representations. We evaluate our method on two downstream tasks: node classification and link prediction. The experimental results demonstrate that the incorporation of vertex and topic embeddings outperform widely-known baseline NRL methods.

1 Introduction

Many pioneer studies in *network representation learning (NRL)* utilize random walks to transform graphs into a collection of sentences – as an analogy to the area of *natural language processing (NLP)* – and these sentences or walks are later being used to learn node embeddings [8]. Although random walk-based approaches are strong enough to capture local patterns of networks, they mainly suffer to sufficiently convey information about the global structural properties of the network. More precisely, real-world networks have an inherent community structure, which can be utilized to further improve the predictive capabilities of node embeddings.

Nevertheless, there are only very few NRL techniques that are benefiting from the community structure of real networks. The authors of [13], have proposed a matrix factorization-based algorithm that incorporates the community structure into the embedding process – implicitly focusing on the quantity of modularity. The ComE model [3] proposes a closed-loop procedure involving the encoding of communities, learning node embeddings and community detection in the network. In our work, we interpret such structural information based on an analogy to the concept of *topics* in a collection of documents. Similar to the domain of NLP where word embeddings can be enhanced with topic-based information [10], here we aim at empowering node embeddings by utilizing information about the community structure of the graph – which 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 and topic embeddings are learned separately from the network, and then are combined into a single vector – leading to further improvements in the performance on downstream tasks. The main contributions of the paper can be summarized as follows:

Relational Representation Learning Workshop (R2L 2018), NIPS 2018, Montréal, Canada.

- *A novel node representation learning framework.* We propose a new strategy, called TNE, which learns community embeddings from the graph, and use them to improve the node representations extracted by random walk-based methods.
- *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 demonstrate, the proposed model provides feature vectors which can boost the performance of downstream tasks.

2 Problem Formulation and Latent Models on Graphs

Our goal is to find a mapping function $\Phi : \mathcal{V} \rightarrow \mathbb{R}^d$, where $\Phi(v)$ indicates the representation of the vertex v in $\mathbb{R}^d$, for $d \ll |\mathcal{V}|$. To learn the representation of a given node, we aim at predicting its nearby nodes using random walks. More specifically, our objective function can be expressed by

$$\mathcal{L}(\Phi, \widetilde{\Phi}) := \max_{\Phi, \widetilde{\Phi}} \sum_{w \in \mathcal{W}} \sum_{v_i \in w} \sum_{-\gamma \leq j \leq \gamma, j \neq 0} \log \mathbb{P}(\Phi(v_{i+j}) | \widetilde{\Phi}(v_i)), \tag{1}$$

where $w = (v_1, \ldots, v_i, \ldots, v_L)$ is a walk of length L, γ refers to the window size, and $\mathcal{W}$ is a collection of walks. Note that, we obtain two different embedding vectors $\Phi(v)$ and $\widetilde{\Phi}(v)$ for each node, but we only consider $\Phi(v)$ as the embedding of the node $v \in \mathcal{V}$. Although some random walk-based methods implicitly benefit from the structural properties of networks, our main goal here is to enhance node embeddings directly by using community structure-based information. We mainly rely on two different approaches to extract latent communities: on random walks and on the network structure itself. In the what follows, we use $\mathcal{K}$ to denote the set of communities of a given graph G.

2.1 Random walk-based graph topic models

When a random walk is initialized, it does not only visit neighboring nodes, but also traverses communities in the network. In this 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. As an analogy to NLP, if each random walk is considered as a document and the collection of random walks as a corpus, the well known Latent Dirichlet Allocation model [4] can be applied to find the topic assignment of each node. (note that, the terms *topic* and *community* will be used interchangeably in the rest of the paper)

By replacing a node v_i with its topic label t_i^w in the walk $\boldsymbol{w}$, we aim to predict the nodes in the context of the topic. More formally, the objective function for topic embeddings will be the following:

$$\mathcal{L}(\Psi, \widetilde{\Psi}) := \max_{\Psi, \widetilde{\Psi}} \sum_{w \in \mathcal{W}} \sum_{v_i \in w} \sum_{-\gamma \leq j \leq \gamma, j \neq 0} \log \mathbb{P}(\Psi(v_{i+j}) | \widetilde{\Psi}(t_i^w)). \tag{2}$$

By maximizing the log-probability of Eq. (2), we obtain the embedding vectors $\widetilde{\Psi}(k) \in \mathbb{R}^d$ for each topic label $k \in \mathcal{K}$, which are called *topic embeddings*. We will refer to this model as *Lda*, and in order to make notations more clear, we will use Φ to refer to node embeddings and $\widetilde{\Psi}$ for topic embeddings.

In the *Lda* model, the latent community assignment of each node is independently chosen from the topic label of the previous node in the walk. However, the hidden state of the current node can play an important role towards determining the next vertex to visit. Hence, we also apply the Bayesian HMM with symmetric Dirichlet priors over transition and emission distributions [5] to detect latent communities. We will refer to this model as *Hmm*.

2.2 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 propose two additional models, namely *BigC* and *Louvain*, which directly target to extract communities of nodes from a given network. The *Louvain* model uses the Louvain community detection method [1], while the *BigC* model is based on an overlapping community detection algorithm called BigClam [15].

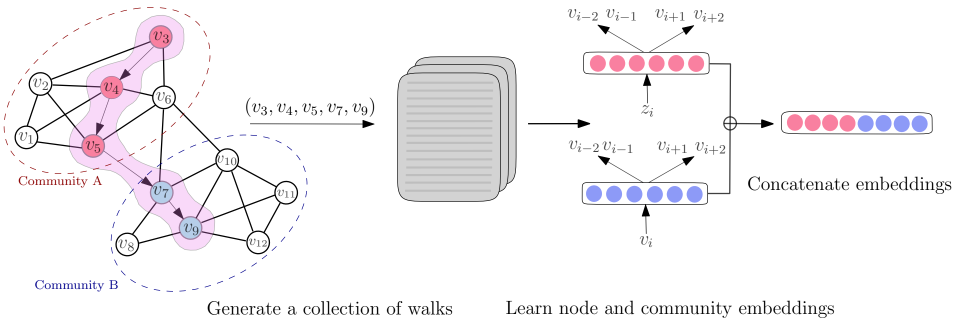

v3
v2
v4
v6
v1
v5
Community A
v10
v7
v11
v9
v8
v12
Community B
(v3, v4, v5, v7, v9)
v_{i−2} v_{i−1} v_{i+1} v_{i+2}
z_i
v_{i−2} v_{i−1} v_{i+1} v_{i+2}
v_i
Concatenate embeddings
Generate a collection of walks
Learn node and community embeddings

Figure 1: Schematic representation of the TNE model.

3 Topical Node Embeddings

Our overall goal is to enhance node embedding using information about the underlying topics of the graph. This can be achieved by learning node and topic embedding vectors independently of each other, jointly maximizing the objectives defined in Equations (1) and (2). By combining these objective functions, we derive the following equation:

$$\mathcal{L}(\Phi, \widetilde{\Phi}, \Psi, \widetilde{\Psi}) := \max_{\Phi, \widetilde{\Phi}, \Psi, \widetilde{\Psi}} \sum_{w \in \mathcal{W}} \sum_{v_i \in w} \sum_{-\gamma \leq j \leq \gamma, j \neq 0} \Big[\log \mathbb{P}(\Phi(v_{i+j}) | \widetilde{\Phi}(v_i)) + \log \mathbb{P}(\Psi(v_{i+j}) | \widetilde{\Psi}(t_i^w)) \Big]$$

An overview of the proposed TNE model is given in Figure 1. First, we need a collection of walks over the network to learn node and topic embeddings. Then, we choose a strategy for this collection to get the topic assignment t_v^w of each node $v \in \mathcal{V}$ in the walk $w \in \mathcal{W}$. This can be achieved based on the *Lda* and *Hmm* generative processes described in the previous section – obtaining the topical node embedding models of *tne-lda* and *tne-hmm*, respectively. Alternatively, the topic assignments can be inferred from the network structure based on the *BigC* or *Louvain* models – relying on the BigClam and Louvain algorithms – and the corresponding topical node embedding models are called *tne-BigC* and *tne-Louvain*.

Afterwards, we produce the node-context pairs to learn node representations $\Phi(v)$, by providing this as input to the Skip-Gram model. By replacing each node v with its topic assignment t_v^w in the walk $w \in \mathcal{W}$, we get a new set of pairs to learn topic embeddings $\widetilde{\Psi}(k)$. Lastly, we obtain the final representation of $v \in \mathcal{V}$ by concatenating its node representation with the embedding vector of the topic k, maximizing the expression $\mathbb{P}(v|k)$.

4 Experimental Evaluation

We apply our framework to the collection of walks generated by the Deepwalk [12] and Node2vec [7] algorithms. Detailed information about the networks used in the experiments is given at [9]. An implementation of our model can be found at: `https://github.com/abdcelikkanat/TNE.git`.

Parameter Settings. The parameters of the random walk methods are assigned to values commonly used in the related literature: $n = 80$, $l = 10$, $\gamma = 10$, and $d = 128$. The hyper-parameters p, q of Node2vec are set to 4.0 and 1.0. To speed up the training process, negative sampling [11] is used for all models with stochastic gradient descent (SGD) [2] for optimization, and the initial learning rate is set to 0.025. Moreover, collapsed Gibbs sampling [6] and variational message passing [14] are used to extract topics for *tne-Lda* and *tne-Hmm* respectively. For all models of the TNE framework, the number of communities is set to $K = 80$ in the experiments.

Multi-Label Node Classification. The goal here is to predict the correct node labels by observing only a certain fraction of the network. Table 1 gives the Macro-F_1 scores, for the case where the size of training and test sets are equal. As it can be seen, *tne-BigC* provides a gain of up to 6.69% compared to the raw Deepwalk model, and up to 6.31% compared to Node2vec on the *Citeseer*

	Citeseer		Cora		PPI	
	Deepwalk	*Node2vec*	*Deepwalk*	*Node2vec*	*Deepwalk*	*Node2vec*
Baseline	0.554	0.551	0.808	0.814	0.174	0.174
tne-Lda	0.590	0.591	0.816	0.822	0.179	0.175
Gain/Loss (%)	**6.58**	**7.32**	**1.04**	**0.96**	**2.83**	**0.47**
tne-Hmm	0.565	0.556	0.807	0.807	0.165	0.164
Gain/Loss (%)	**2.02**	**0.84**	**-0.03**	**-0.93**	**-5.01**	**-5.68**
tne-BigC	0.591	0.586	0.814	0.817	0.168	0.169
Gain/Loss (%)	**6.69**	**6.31**	**0.81**	**0.28**	**-3.14**	**-2.90**
tne-Louvain	0.589	0.593	0.819	0.823	0.175	0.173
Gain/Loss (%)	**6.45**	**7.58**	**1.42**	**1.10**	**0.80**	**-0.47**

Table 1: Macro-F_1 scores for node classification, where 50% of nodes are used for training. It shows the performance of the different TNE models applied on walks extracted by Deepwalk and Node2vec.

		(a)		(b)		(c)		(d)	
		Deepwalk	Node2vec	Deepwalk	Node2vec	Deepwalk	Node2vec	Deepwalk	Node2vec
Gnutella	*Baseline*	0.5952	0.5944	0.7050	0.7148	0.5825	0.6194	0.5790	0.6171
	tne-Lda	0.5920	0.5991	0.7043	0.7086	0.5852	0.6224	0.5820	0.6208
	tne-Hmm	0.5961	0.5916	**0.7125**	**0.7261**	0.5821	0.6179	0.5732	0.6137
	tne-BigC	0.5988	**0.6017**	0.7047	0.7227	0.5863	**0.6272**	0.5804	**0.6256**
	tne-Louvain	**0.5998**	0.5945	0.6991	0.7071	**0.5873**	0.6188	**0.5827**	0.6158
Facebook	*Baseline*	0.9862	0.9865	0.7537	0.7505	0.9839	0.9831	0.9840	0.9834
	tne-Lda	0.9882	0.9888	0.7772	0.7749	0.9859	0.9861	0.9861	0.9866
	tne-Hmm	**0.9884**	0.9884	**0.7789**	**0.7784**	0.9864	0.9860	0.9868	0.9862
	tne-BigC	0.9882	**0.9890**	0.7715	0.7731	**0.9869**	**0.9864**	**0.9870**	**0.9867**
	tne-Louvain	0.9881	0.9888	0.7597	0.7615	0.9846	0.9842	0.9847	0.9845
ArXiv gr-qc	*Baseline*	0.9262	0.9314	0.7256	0.7254	0.9249	0.9304	0.9253	0.9312
	tne-Lda	**0.9328**	0.9346	0.7232	0.7249	**0.9335**	**0.9319**	**0.9337**	0.9323
	tne-Hmm	0.9220	0.9332	0.7223	0.7290	0.9207	0.9304	0.9212	0.9321
	tne-BigC	0.9271	0.9309	**0.7273**	0.7311	0.9237	0.9288	0.9228	0.9294
	tne-Louvain	0.9302	**0.9353**	0.7320	**0.7375**	0.9274	0.9340	0.9266	**0.9342**

Table 2: Area Under Curve (AUC) scores for the link prediction task with four different binary operators: (a) Hadamard, (b) Average, (c) Weighted-L1, and (d) Weighted-L2. The first row of each block corresponds to the raw performance of Deepwalk and Node2vec.

dataset. Although the general performance of Node2vec and Deepwalk are the same on the PPI network, the proposed *tne-Lda* model offers a gain of 2.83%.

Link Prediction. For the link prediction task, 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 that does not exist in the initial network is chosen to obtain negative samples for each training and test sets. Table 2 presents the area under curve (AUC) scores. As it can be observed, the proposed models outperform the baseline methods in all cases. For the *Facebook* network, *tne-BigC* gives the best results for all but the average operator – which also corresponds to one of the best performing model across all different settings.

5 Conclusions and Future Work

In this paper, we proposed TNE, a latent model for representation learning on networks. By taking advantage of the underlying community structure of the network, TNE is capable of producing enriched latent node representations, compared to traditional random walk-based approaches, leading to improved performance results in the tasks of node classification and link prediction. Currently, TNE can only be applied along with random walk-based approaches. An interesting future direction is how to extend the framework to include other NRL algorithms. Moreover, motivated by the hierarchical community structure that many real networks follow, an interesting future direction would be to extend the framework towards learning hierarchical node embeddings. Lastly, we plan to evaluate TNE in the task of community detection.

References

[1] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech.*, 2008(10):P10008, 2008.

[2] Léon Bottou. Stochastic gradient learning in neural networks. In *Proceedings of Neuro-Nîmes, EC2*, 1991.

[3] Sandro Cavallari, Vincent W. Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. Learning community embedding with community detection and node embedding on graphs. In *CIKM*, pages 377–386, 2017.

[4] Michael I. Jordan David M. Blei, Andrew Y. Ng. Latent dirichlet allocation. *Journal of Machine Learning Research.*, 3(1):993–1022, January 2003.

[5] Sharon Goldwater and Tom Griffiths. A fully bayesian approach to unsupervised part-of-speech tagging. In *ACL*, pages 744–751, 2007.

[6] Thomas L. Griffiths and Mark Steyvers. Finding scientific topics. *PNAS*, 101(suppl 1):5228–5235, 2004.

[7] Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864, 2016.

[8] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*, 40(3):52–74, 2017.

[9] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[10] Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. Topical word embeddings. In *AAAI*, pages 2418–2424, 2015.

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[12] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD*, pages 701–710, 2014.

[13] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *AAAI*, pages 203–209, 2017.

[14] John Winn and Christopher M. Bishop. Variational message passing. *J. Mach. Learn. Res.*, 6:661–694, December 2005.

[15] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: A nonnegative matrix factorization approach. In *WSDM*, pages 587–596, 2013.

Appendix

Network statistics

The statistics of the networks used in the node classification and link prediction experiments are given in Table 3. The networks are converted into undirected, in order to ensure the consistency of the experiments.

Name	Citeseer	Cora	PPI	Gnutella	Facebook	ArXiv gr-qc
# Vertices	3,312	2,708	3,890	8,114	4,039	5,242
# Edges	4,660	5,278	38,739	26,013	88,234	14,496
# Clusters	6	7	50	-	-	-

Table 3: Statistics of the networks used in the experiments.

Multi-label node classification

Figure 2 depicts the Micro-F_1 scores for the different variants of the TNE framework applied on walks generated by Deepwalk (first row) and Node2vec (second row), as well as the performance of those methods itself (denoted by *deepwalk-emb* and *node2vec-emb*).

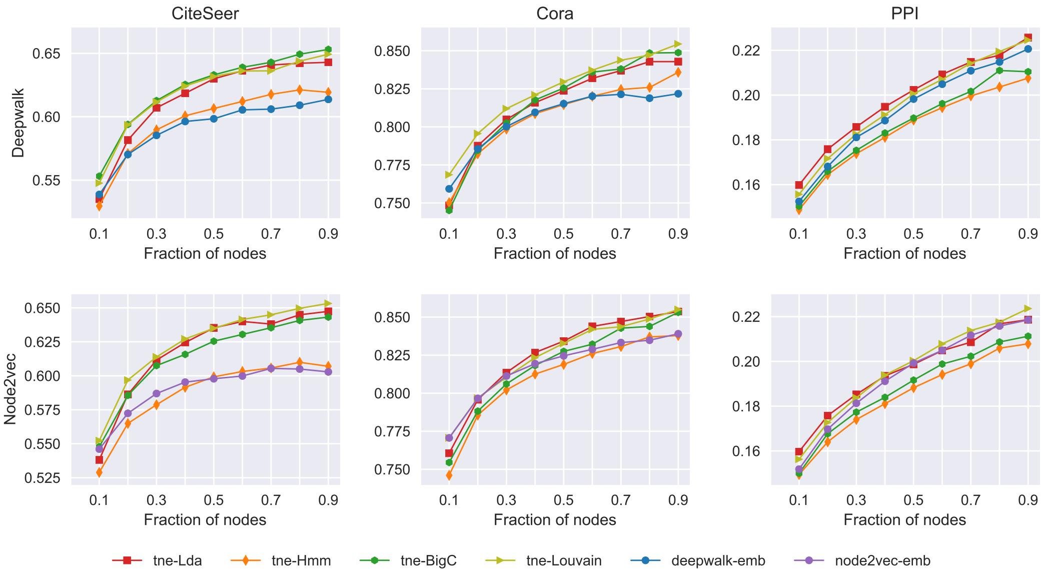
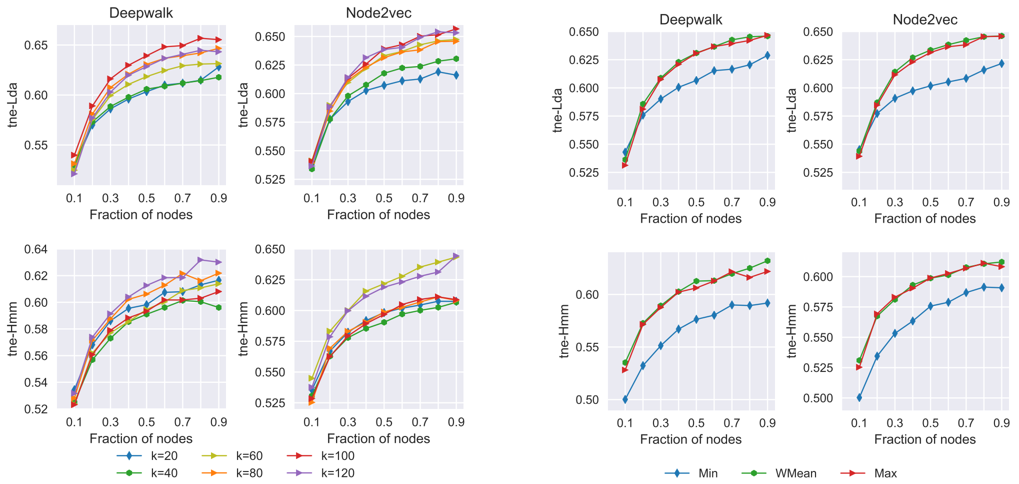


Figure 2: Performance evaluation of the proposed TNE framework against Deepwalk and Node2vec, over a varying fraction of training data. The x-axis indicates the ratio of the training dataset, and the y-axis shows the Micro-F_1 scores for different random walk strategies on three different networks.

The effect of the number of topics

We analyze the effect of the number of topics (or clusters) in the performance of our framework. The experiments are performed on the CiteSeer network and we examine the *tne-Lda* and *tne-Hmm* models on the collection of random walks generated by Deepwalk and Node2vec. Figure 3a indicates that the increase in the number of topics makes positive contribution up to a certain value for the *tne-Lda* model. On the other hand, this is not valid for *tne-Hmm*; it performs better for $K = 120$ over both random walk strategies. The chosen number of topics shows its importance for large training data sizes – the scores get closer to each other when the training size decreases.



(a) Micro-F_1 scores for various values of the number of topics for the CiteSeer network.

(b) The effect of different embedding concatenation methods for the CiteSeer network.

Figure 3: Performance evaluation with respect to the number of topics and the different embedding concatenation strategies in multi-label classification.

The effect of the concatenation strategy

We define three different concatenation strategies to obtain the final representation vector of each node. For the Max and Min methods, the node representation vector $\Phi(v)$ is concatenated with the embedding $\widetilde{\Psi}(k)$ of the topic k maximizing and minimizing the expression $\Pr(\Psi(v)|\widetilde{\Psi}(k))$, respectively. For $WMean$, the strategy is formulated as follows: $WMean(v) := \Phi(v) \oplus \sum_k \widetilde{\Psi}(k) \cdot \mathbb{P}(v|k)$.

Here, we perform several experiments to observe the behavior of those strategies over varying training data sizes. Figure 3b depicts the Micro-F_1 scores on the CiteSeer network. As it can be seen, the Max and $WMean$ strategies highly outperform the third one across all cases, and their scores are highly close to each other. We have preferred to use the Max concatenation strategy in all our experiments.