Network Embedding by Walking on the Line Graph

Miguel Angel Lozano[1], Manuel Curado[1], Francisco Escolano[1](✉), and Edwin R. Hancock[2]

[1] Department of Computer Science and AI, University of Alicante, Alicante, Spain
malozano@ua.es, {mcurado,sco}@dccia.ua.es
[2] Department of Computer Science, University of York, York, UK
edwin.hancock@york.ac.uk

Abstract. In this paper, we propose to embed edges instead of nodes using state-of-the-art neural/factorization methods (DeepWalk, node2vec). These methods produce latent representations based on co-ocurrence statistics by simulating fixed-length random walks and then taking bags-of-vectors as the input to the Skip Gram Learning with Negative Sampling (SGNS). We commence by expressing commute times embedding as matrix factorization, and thus relating this embedding to those of DeepWalk and node2vec. Recent results showing formal links between all these methods via the spectrum of graph Laplacian, are then extended to understand the results obtained by SGNS when we embed edges instead of nodes. Since embedding edges is equivalent to embedding nodes in the line graph, we proceed to combine both existing formal characterizations of the line graphs and empirical evidence in order to explain why this embedding dramatically outperforms its nodal counterpart in multi-label classification tasks.

Keywords: Network embedding · SGNS · Line graph · Spectral theory

1 Introduction

The recent success of neural graph embeddings such as LINE [18], DeepWalk [14] and node2vec [7] has opened a new path for analyzing networks. Despite these embeddings outperform spectral ones in tasks such as link prediction and multi-label node classification, Spectral Graph Theory [3] is still key tool for understanding and characterizing neural embeddings [16].

In this paper, we contribute with empirical evidence showing that neural embeddings (Sect. 2) can boost their performance in multi-label classification by embedding edges instead of nodes. We conjecture that this fact is due to the spectral properties of *line graphs*, whose nodes are the edges of the original graphs

M. A. Lozano, M. Curado and F. Escolano are funded by the project TIN2015-69077-P of the Spanish Government.

(Sect. 3). However, since general line graphs have not been fully characterized yet, we can only correlate our empirical findings (Sect. 4) with some of the well known properties of line graphs and the spectral characterization of neural embeddings.

2 Classic vs Neural Embeddings

2.1 Classic Embeddings

Let $G = (V, E, \mathbf{A})$ be a graph/network with $n = |V|$ nodes, $m = |E|$ edges, where $E \subseteq V \times V$, and adjacency matrix $\mathbf{A}$. Then, *node embedding* consists of finding a mapping $f : V \rightarrow \mathbb{R}^d$ (with $d \ll n$) so that the resulting d-dimensional vectors capture the structural properties of each vertex. As a result, we have $||f(i) - f(j)||^2 \rightarrow 0$ if nodes i and j are structurally similar within the graph G. Traditionally, nodal structural similarity was associated with the reachability of node j from node i (and vice versa) through random walks [10]. This characterization leaded to define both *hitting times* H_{ij} (expected steps taken by a random walk to reach j from i) and *commute times* $CT_{ij} = H_{ij} + H_{ji}$ (which also includes the expected steps needed to return to i from j). Since random walks are encoded by transition matrices of the form $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$, where $\mathbf{D} = diag(d_1, \ldots, d_n)$ is the diagonal matrix with the degrees of the nodes, the spectral analysis of $\mathbf{P}$ is a natural way of understanding both hitting and commute times. More precisely, let $\lambda_1 = 1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq -1$ be the spectrum of the transition matrix. It is well known that hitting times and commute times are highly conditioned by the *spectral gap* $\lambda = 1 - \max\{\lambda_2, |\lambda_n|\}$. When several communities are encoded by a connected graph G, then H_{ij} and CT_{ij} are only meaningful when $\lambda \rightarrow 0$ (small bottlenecks between communities); otherwise, these quantities rely on the local densities (degrees) of the nodes i and j, and one cannot discriminate whether two nodes belong to the same community or not [11]. Consequently, the applicability of node embeddings based on commute times to clustering is quite limited (see representative examples of image segmentation and tracking in [15]). In this regard, recent research is focused on simultaneously minimizing the spectral gap and shrinking (whenever possible) inter-community commute distances via graph densification [4,5] before embedding the nodes.

Therefore, once G is processed (or rewired) commute times embedding leads to learn two matrices $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{n \times d}$, whose rows are denoted by $\mathbf{x}_i$ and $\mathbf{y}_i$ respectively and $\mathbf{x}_i$ is the embedding of the node i. Following [15], the commute times embedding matrix $\mathbf{X}$ results from factorizing

$$vol(G)\mathcal{G} = \mathbf{X}\mathbf{Y}^T, \tag{1}$$

where $vol(G) = \sum_{i=1}^{n} d_i$ is the volume of the graph and $\mathcal{G}$ is its Green's function, i.e. the pseudo-inverse of the normalized graph Laplacian $\mathcal{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, whose spectrum is $1 - \lambda_1 = 0, 1 - \lambda_2, \ldots, 1 - \lambda_n \leq 2$, i.e. if λ_i is an eigenvalue of $\mathbf{P}$ then $1 - \lambda_i$ is an eigenvalue of $\mathcal{L}$.

2.2 Neural Embeddigs

Neural embeddings such as LINE [18], DeepWalk [14] and node2vec [7], exploit random walks in a different way. Namely, they simulate a fixed number N of random walks with fixed length L emanating from the nodes of G and then capture co-ocurrence statistics of pairs of nodes. The first node of the i-th path w_i, assimilated to a word in a textual corpus (skip-gram model), is sampled from a prior distribution $P(w_i)$. Then, the context of w_i is given by the nodes/words surrounding it in a T-sized window $w_{i-T}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+T}$, according to the transition matrix $\mathbf{P}$. Then, the node-context pairs (w, c) are given by (w_{i-r}, w_i) and (w_i, w_{i+r}) for $r = 1, \ldots, T$. All these pairs are added to the multiset $\mathcal{D}$ used for learning with *negative sampling*. Negative sampling implies not only to consider likely node-context pairs (w, c) but also b unlikely ones (w, c'): the negative samples c', are nodes that can be drawn from the steady-state probability distribution of the random walk, i.e. $P_N(i) = d_i/vol(G)$. This process is called Skip Gram Learning with Negative Sampling (SGNS) and leads to the following factorization [9]:

$$\mathbf{M} = \mathbf{X}\mathbf{Y}^T, \text{ with } \mathbf{M}_{ij} = \log\left(\frac{\#(w_i, c_j)|\mathcal{D}|}{\#(w_i)\#(c_j)}\right) - \log b, \tag{2}$$

where: $\#(w_i, c_i)$ is the number of times the corresponding node-context pair is observed,$\#(w_i)$ is the number of times the node i is observed and similarly for node $\#(c_j)$; finally $\log(.)$ is the element-wise logarithm and b is the number of negative samples.

2.3 LINE and DeepWalk vs node2vec Factorizations

These strategies differ in the way they sample (and thus vectorize) the graph for SGNS. LINE and DeepWalk rely on first-order random walks whereas node2vec is driven by second-order random walks.

LINE and DeepWalk. LINE's factorization is a direct result from the cost function associated with SGNS. In particular, the latent representations of both the word/node $\mathbf{x}_i$ and the context $\mathbf{y}_j$ are assumed to be correlated with the existence on an edge between nodes i and j, i.e. $\mathbf{A}_{ij} \log g(\mathbf{x}_i^T \mathbf{y}_j)$ is maximized, where $g(.)$ is the sigmoid function. Following [16], this leads to

$$\mathbf{x}_i^T \mathbf{y}_j = \log\left(\frac{vol(G)\mathbf{A}_{ij}}{d_i d_j}\right) - \log b \;\Rightarrow \log\left(vol(G)\mathbf{D}^{-1}\mathbf{A}\mathbf{D}^{-1}\right) - \log b = \mathbf{X}\mathbf{Y}^T. \tag{3}$$

DeepWalk, on the other hand, leads to a more complex factorization. Assuming that the first node of each random walk is drawn from the steady state distribution, we have that, when $L \to \infty$,

$$\frac{\#(w_i, c_j)|\mathcal{D}|}{\#(w_i)\#(c_j)} \xrightarrow{p} \frac{vol(G)}{2T}\left(\frac{1}{d_j}\sum_{r=1}^{T}\mathbf{P}^r_{ij} + \frac{1}{d_i}\sum_{r=1}^{T}\mathbf{P}^r_{ji}\right) \tag{4}$$

where $\xrightarrow{p}$ denotes *convergence in probability*. This yields

$$\log\left(\frac{vol(G)}{T}\left(\sum_{r=1}^{T}\mathbf{P}^r\right)\mathbf{D}^{-1}\right)-\log b=\mathbf{X}\mathbf{Y}^T, \tag{5}$$

which is equivalent to LINE for $T=1$.

node2vec. The underlying idea of this embedding is to add more flexibility to the random walk. This is done by defining two parameters p and q, that control, respectively the likelihood of immediately revisit a node in the walk and making the walk very local. To that end, node2vec needs to evaluate the probability of the next nodes given the preceding one in the walk, i.e. we have a 2nd-order random walk. This walk is characterized by the hypermatrix $\underline{\mathbf{P}}$, where $\underline{\mathbf{P}}_{i(jk)}$ denotes the probability of reaching j from j given that the node preceeding j is k. Thus, the 2nd order random walk can be reduced to a 1st order one on the edges of the graph [1] as it is done in the implementation of node2vec. The stationary distribution $\mathbf{X}_{ik}$ for this type of random walks satisfies $\sum_k \underline{\mathbf{P}}_{i(jk)}\mathbf{X}_{ik}=\mathbf{X}_{ij}$. Qiu et al. [16] have found that

$$\frac{\#(w_i,c_j)|\mathcal{D}|}{\#(w_i)\#(c_j)}\xrightarrow{p}\frac{\frac{1}{2T}\sum_{r=1}^{T}\left(\sum_k \mathbf{X}_{ik}\underline{\mathbf{P}}^r_{j(ik)}+\sum_k \mathbf{X}_{jk}\underline{\mathbf{P}}^r_{i(jk)}\right)}{\left(\sum_k \mathbf{X}_{ik}\right)\left(\sum_k \mathbf{X}_{jk}\right)} \tag{6}$$

and, despite the matricial expression for the factorization is more elusive, the final factorization differs significantly from those of DeepWalk and LINE.

3 Node vs Edges Embedding

3.1 The Line Graph

In this paper, we are mainly concerned with the impact of embedding the edges of G instead of its nodes. This means that a word w_i in the previous expressions is not yet associated with a node of G but with a node of its *line graph* L_G. The nodes of L_G are the edges of G and there is an edge in the line graph if two edges in G share a node. More formally, given the $n\times m$ incidence matrix $\mathbf{B}$ where $\mathbf{B}_{i\alpha}$ is 1 if the link α is related to node i and 0 otherwise, we have that the $m\times m$ adjacency matrix $\mathbf{C}=\mathbf{B}^T\mathbf{B}-\mathbf{I}$ has elements $C_{\alpha\beta}=\sum_{i=1}^{n}\mathbf{B}_{i\alpha}\mathbf{B}_{i\beta}(1-\delta_{\alpha\beta})$.

3.2 Spectral Analysis

Some interesting properties of line graphs vs G:

- *Boosted edge density.* A single node i in G leads to a clique of $d_i(d_i-1)/2$ edges in L_G (see Fig. 1). Despite this gives a high prominence to notable nodes of G it flexibilizes community detection [6]. In addition the steady state distribution of a random walk in L_G is $P_N(\alpha_{(i,j)})=d_\alpha/vol(L_G)$ where $d_\alpha=d_i+d_j-2$ and $vol(L_G)=\sum_{\alpha,\beta}C_{\alpha\beta}=\sum_{i=1}^{n}d_i(d_i-1)$.

- *Redundant spectrum for $m > n$.* Let $\lambda_1(L_G) \geq \lambda_2(L_G) \geq \ldots \geq \lambda_m(L_G)$ be the spectrum of $\mathbf{C}$. Then, for $m > n$, $\lambda_{n+1} = \ldots = \lambda_m = -2$. As a result, $\lambda_i(\mathbf{L}(L_G)) \geq 4$, for the largest $m - n$ eigenvalues of $\mathbf{L}(L_G)$, the unnormalized Laplacian matrix of L_G [19]. This increases significantly the medium-large eigenvalues of $\mathcal{L}(L_G)$ with respect to $\mathcal{L}(G)$ (see Fig. 2).
- *Majorization of the spectrum of G.* This is really a conjecture derived from the bound $\lambda_2(L_G) \leq \frac{m}{2} - 1$ in comparison to that for G: $\lambda_2(G) \leq \frac{n}{2} - 1$. Empirical data shows that the lowest part of the spectrum of $\mathcal{L} - \mathbf{I}$ in the line graph majorizes that of G (blue lines in Fig. 2). Since the spectrum driving DeepWalk is (approximately) of the form $\frac{1}{T}\sum_{r=1}^{T}\lambda_i^r$ this leads (in general) to small spectral gaps for the line graphs, and thus slower mixing times of the random walks (more randomness). Green lines show the real spectra driving random walks in DeepWalk. In all cases, $T = 10$.

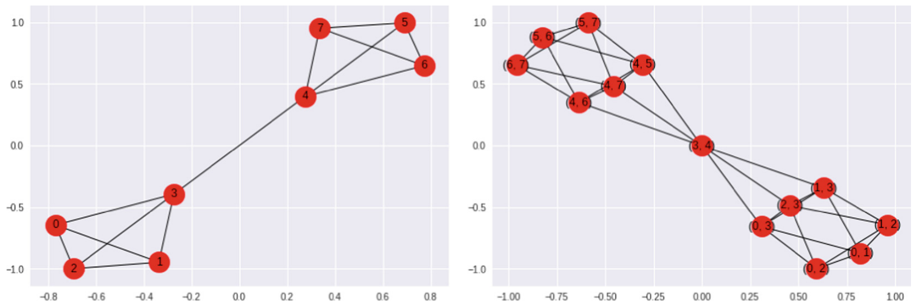

Fig. 1. Barbell graph linking two cliques (left) and its line graph (right)

4 Experiments and Discussion

4.1 Datasets (Networks)

- **CiteSeer for Document Classification** [17]. Citation network containing 3312 scientific publications with 4676 links between them. Each publication is classified in one of 6 categories.
- **Cora** [17]. Citation network containing 2708 scientific publications with 5278 links between them. Each publication is classified in one of 7 categories.
- **Wiki**[1]. Contains a network of 2405 web pages with 17981 links between them. Each page is classified in one of 19 categories.
- **Facebook** social circles [13].

[1] https://github.com/thunlp/MMDW/.

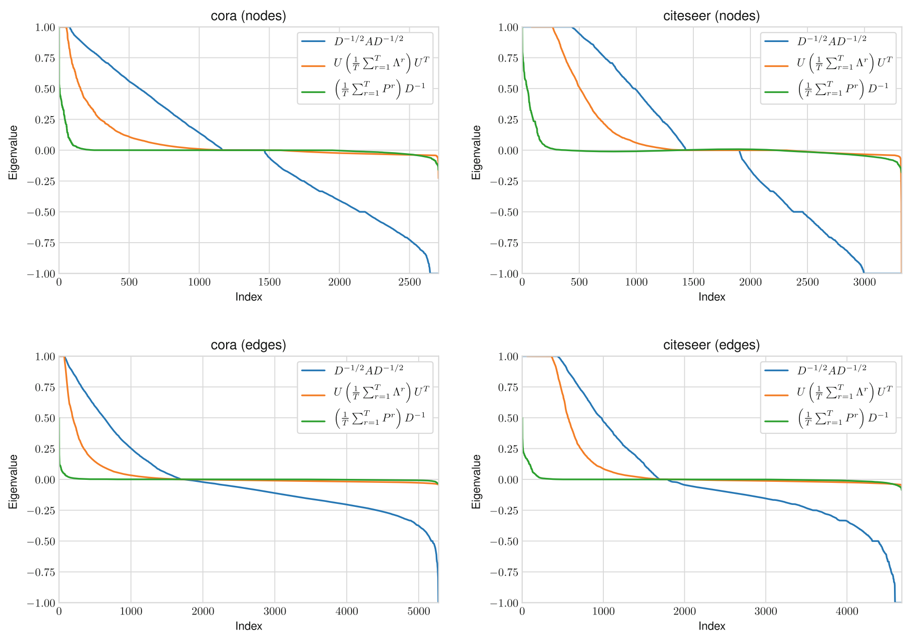

Fig. 2. Eigenvalues of the original (top) and line graph (bottom), for Cora (left) and CiteSeer (right) databases (Color figure online)

	Nodes	Edges	Line graph edges	Gap	Con. comps	Labels	Multi-label
wiki	2405	12761	355644	0.000000	45	19	No
cora	2708	5278	52301	0.000000	78	7	No
citeseer	3327	4676	27174	0.000000	438	6	No
ppi	3890	38739	3018220	0.000000	35	50	Yes
pos	4777	92517	49568882	0.576132	1	40	Yes
facebook	4039	88234	9314849	0.000837	1	10	Yes

- **Wikipedia Part-of-Speech (POS)** [12]. Co-ocurrence of words appearing in the first million of the bytes of the dumping of Wikipedia. The categories correspond to the labels of Part-of-Speech (POS) inferred by the Stanford POS-Tagger. Contains 4777 nodes, and 92517 undirected links. Each node may have several labels. We have 40 labels (categories).
- **Protein-Protein Interactions (PPI)**[2] [2]. We use a subgraph of the PPIs associated with the Homo Sapiens. The network has 3890 nodes and 76584 links. Each node may have several labels corresponding to the 50 possible categories.

[2] https://downloads.thebiogrid.org/BioGRID.

Facebook, PPI and POS have been retrieved from SNAP[3] [8]. CiteSeer and Cora have been retrieved from LINQS[4].

All the networks are considered as undirected graphs. Originally single-labelled networks are transformed into multi-label networks when their line-graph is computed (or sampled, for the sake of efficiency). Nodes with more than one label are the *border nodes* between two categories (inter-class), and the nodes that hold one label are intra-class nodes.

	Inter-class nodes	Intra-class nodes
wiki	4526 (35%)	8235 (65%)
cora	1003 (19%)	4275 (81%)
citeseer	1190 (25%)	3486 (75%)

We have used the implementations of node2vec and DeepWalk included in the framework. OpenNE[5]. The default values for p and q in node2vec are $p = q = 1$. After optimizing p and q in the range $\{0.25, 0.5, 1, 2, 4\}$ the maximum improvement of node2vec wrt DeepWalk in the classification score is 0.014 (micro and macro). Regarding spectral embeddings, CTE and LLE have been only tested in networks with a single connected component (pos and facebook). In particular, commute times (CTE) have a poor performance in multi-label classification because their factorization relies on the Green's function and this means that only the inverse of each eigenvalue is considered. However, DeepWalk is controlled by a polynomial associated with each eigenvalue.

		node2vec		DeepWalk		cte	lle
		Nodes	Edges	Nodes	Edges		
Micro-F1	citeseer	0.591071	0.768626	0.595833	**0.780930**	–	–
	cora	0.808715	0.903557	0.817578	**0.920538**	–	–
	wiki	0.663342	0.840709	0.692436	**0.859129**	–	–
	pos	0.447845	**0.697572**	0.471620	0.696640	0.375037	0.393165
	ppi	0.197435	0.590731	0.205786	**0.609937**	–	–
	facebook	0.911516	**0.999900**	0.911516	**0.999900**	0.239217	0.231214

[3] https://snap.stanford.edu/node2vec/.
[4] https://linqs.soe.ucsc.edu/data.
[5] https://github.com/thunlp/OpenNE.

		node2vec		DeepWalk		cte	lle
		Nodes	Edges	Nodes	Edges		
Macro-F1	citeseer	0.544490	0.730930	0.545774	**0.745995**	–	–
	cora	0.798782	0.898863	0.804928	**0.917838**	–	–
	wiki	0.528603	0.764205	0.597948	**0.787771**	–	–
	pos	0.084183	0.773035	0.094148	**0.774001**	0.041637	0.033617
	ppi	0.168237	0.566912	0.178405	**0.587934**	–	–
	facebook	0.821899	0.999377	0.822243	**0.999407**	0.108742	0.045155

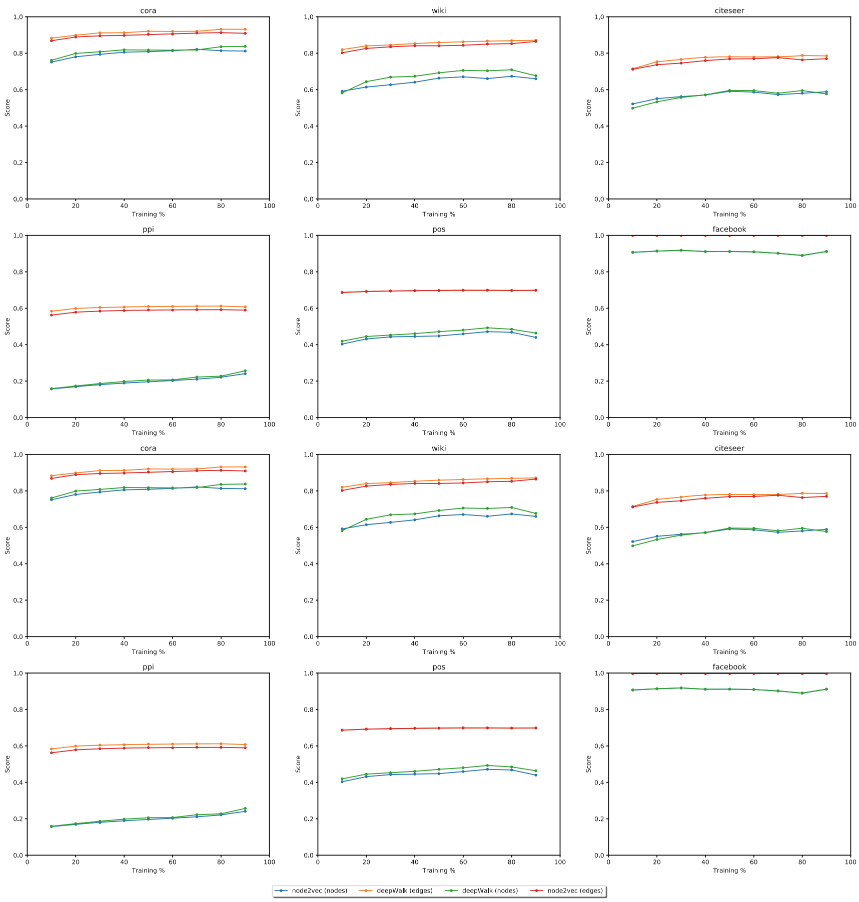

Fig. 3. Evolution of the performance as a function of the fraction of known labels in the training set. Micro-F1 (top) and Macro-F1 (bottom)

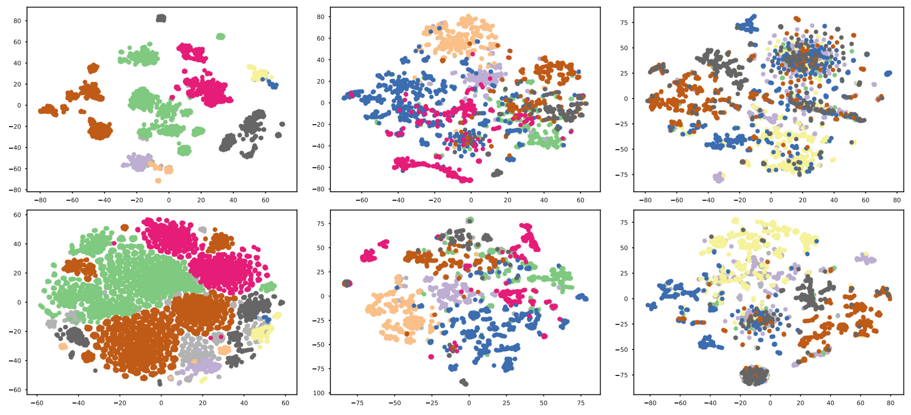

Fig. 4. t-SNE embeddings. Original graph (top) and line-graph (bottom), for Facebook (left), Cora (center) and CiteSeer (right) databases.

In Fig. 3, we show the performance in multi-label classification (according to the percentage of nodes with known labels). The line-graph versions of node2vec and DeepWalk clearly outperform their nodal counterparts. The similarity in terms of performance of node2vec and DeepWalk is due to the fact that the 2nd order random walk of node2vec is not applied at the level of edges (it is unfeasible for large networks). Finally, in Fig. 4 we show the t-SNE embeddings. Edge embeddings clearly produce denser communities.

5 Conclusions

In this paper, we have contributed with empirical evidence showing that embedding edges clearly outperforms node-based embeddings in neural SGNS strategies. We conjecture that this is due to the slower mixing times of random walks in line graphs. Future work includes a detailed check of this conjecture as well as more efficient (in time and space) strategies for designing walkers on the line graphs.

References

1. Benson, A.R., Gleich, D.F., Lim, L.: The spacey random walk: a stochastic process for higher-order data. SIAM Rev. **59**(2), 321–345 (2017). https://doi.org/10.1137/16M1074023
2. Breitkreutz, B., et al.: The biogrid interaction database: 2008 update. Nucleic Acids Res. **36**, 637–640 (2008). https://doi.org/10.1093/nar/gkm1001
3. Chung, F.R.K.: Spectral graph theory. In: Conference Board of the Mathematical Sciences (CBMS), number 92. American Mathematical Society (1997)

4. Curado, M., Escolano, F., Lozano, M.A., Hancock, E.R.: Dirichlet densifiers: beyond constraining the spectral gap. In: Bai, X., Hancock, E., Ho, T., Wilson, R., Biggio, B., Robles-Kelly, A. (eds.) S+SSPR 2018. LNCS, vol. 11004, pp. 512–521. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-97785-0_49
5. Escolano, F., Curado, M., Lozano, M.A., Hancook, E.R.: Dirichlet graph densifiers. In: Robles-Kelly, A., Loog, M., Biggio, B., Escolano, F., Wilson, R. (eds.) S+SSPR 2016. LNCS, vol. 10029, pp. 185–195. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49055-7_17
6. Evans, T.S., Lambiotte, R.: Line graphs, link partitions, and overlapping communities. Phys. Rev. E **80**, 016105 (2009). https://doi.org/10.1103/PhysRevE.80.016105
7. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, 13–17 August 2016, pp. 855–864 (2016). https://doi.org/10.1145/2939672.2939754
8. Leskovec, J., Krevl, A.: SNAP datasets: Stanford large network dataset collection, June 2014. http://snap.stanford.edu/data
9. Levy, O., Goldberg, Y.: Neural word embedding as implicit matrix factorization. In: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems, 8–13 December 2014, Montreal, pp. 2177–2185 (2014). http://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization
10. Lovász, L.: Random walks on graphs: a survey. In: Miklós, D., Sós, V.T., Szőnyi, T. (eds.) Combinatorics, Paul Erdős is Eighty, vol. 2, pp. 353–398. János Bolyai Mathematical Society, Budapest (1996)
11. von Luxburg, U., Radl, A., Hein, M.: Hitting and commute times in large random neighborhood graphs. J. Mach. Learn. Res. **15**(1), 1751–1798 (2014). http://dl.acm.org/citation.cfm?id=2638591
12. Mahoney, M.: Large text compression benchmark (2011). http://www.mattmahoney.net/dc/textdata
13. McAuley, J.J., Leskovec, J.: Learning to discover social circles in ego networks. In: Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems, Proceedings of a Meeting Held 3–6 December 2012, Lake Tahoe, pp. 548–556 (2012). http://papers.nips.cc/paper/4532-learning-to-discover-social-circles-in-ego-networks
14. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014, New York, 24–27 August 2014, pp. 701–710 (2014). https://doi.org/10.1145/2623330.2623732
15. Qiu, H., Hancock, E.R.: Clustering and embedding using commute times. IEEE Trans. Pattern Anal. Mach. Intell. **29**(11), 1873–1890 (2007). https://doi.org/10.1109/TPAMI.2007.1103
16. Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., Tang, J.: Network embedding as matrix factorization: unifying DeepWalk, LINE, PTE, and node2vec. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, pp. 459–467. ACM, New York (2018). https://doi.org/10.1145/3159652.3159706

17. Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. AI Mag. **29**(3), 93–106 (2008). http://www.aaai.org/ojs/index.php/aimagazine/article/view/2157
18. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, 18–22 May 2015, pp. 1067–1077 (2015). https://doi.org/10.1145/2736277.2741093
19. Yan, C.: Properties of spectra of graphs and line graphs. Appl. Math. J. Chin. Univ. **17**(3), 371–376 (2002). https://doi.org/10.1007/s11766-002-0017-7