Influence of Random Walk Parametrization on Graph Embeddings

Fabian Schliski^(✉), Jörg Schlötterer, and Michael Granitzer

University of Passau, 94032 Passau, Germany

`schliski@fim.uni-passau.de`,

`{joerg.schloetterer,michael.granitzer}@uni-passau.de`

Abstract. Network or graph embedding has gained increasing attention in the research community during the last years. In particular, many methods to create graph embeddings using random walk based approaches have been developed. node2vec [10] introduced means to control the random walk behavior, guiding the walks. We aim to reproduce parts of their work and introduce two additional modifications (jump probabilities and attention to hubs), in order to investigate how guiding and modifying the walks influences the learned embeddings. The reproduction includes the case study illustrating homophily and structural equivalence subject to the chosen strategy and a node classification task. We were not able to illustrate structural equivalence and further results show that modifications of the walks only slightly improve node classification, if at all.

Keywords: Feature learning · Graph embedding · Random walk

1 Introduction

Network analysis involves methods to predict over nodes and edges, such as node classification [5], link prediction [12], clustering [8], and visualization [13].

Node classification aims at predicting the labels of unlabeled nodes based on a set of different labeled nodes and the network topology. An example is to predict the interests of a user in a social network based on other users with overlapping characteristics. Link prediction is used to predict missing or future links between nodes in the network. In a social network, it can be used to recommend new friends based on the current ones. Clustering attempts to identify similarities between nodes in the network and groups them into same-labeled clusters. This can be used to detect communities with similar interests in a social network. Visualization helps to gain quick insights about the structure of the network.

Graph embeddings are feature vector representations of the nodes and edges of a network and are used as input to the methods above. node2vec [10] uses a random-walk based approach to create those embeddings and introduces two additional parameters to guide the random walk, aiming at preserving both community structure and structural roles. Community structure in a graph is

based on proximity, i.e., nodes that are close together belong to a community. Structural roles can be nodes that act as bridges between sub-networks, or hubs, which are the main exchange point between many nodes. The two parameters guiding the random walk in node2vec [10] are:

- Return parameter p , controlling the likeliness of immediately revisiting a node.
- In-out parameter q , controlling how far outward the random walk should progress from the starting node.

These parameters allow to resemble depth-first (DFS) or breadth-first (BFS) search-like behaviour in the most extreme setting, as well as a smooth interpolation between DFS and BFS. Grover and Leskovec suggest “that BFS and DFS strategies represent extreme ends on the spectrum of embedding nodes based on the principles of homophily (i.e., network communities) and structural equivalence (i.e., structural roles of nodes)” [10]. In a case study, they demonstrate that subject to the parameter settings, the resulting embeddings can capture homophily or structural equivalence. In this paper¹, we

- try to reproduce the case study illustrating homophily and structural equivalence.
- try to reproduce node2vec’s node classification result.
- introduce two additional modifications to the random walk strategy and evaluate and compare them on the node classification task. The additional strategies comprise hub attention and jump probabilities, where the latter can be seen as noise.

2 Related Work

Algorithms to create graph embeddings can be divided into three categories: Factorization based, deep learning based, and random walk based [9].

Factorization based algorithms represent the graph as a matrix and apply methods such as eigenvalue decomposition or gradient descent to obtain node embeddings [9]. Examples are LLE [19], Laplacian Eigenmaps [4], GraRep [6], and HOPE [16].

The deep learning based methods try to improve the performance of the factorization algorithms by computing non-linear functions on the graph. Examples are SDNE [20] (auto-encoder to reduce dimensionality), DNGR [7] (deep neural networks), and GCN [11] (graph convolutional networks).

Random walk based approaches create embeddings by processing sets of random walks through the graph. First in this line was DeepWalk [18], which samples purely random walks. These walks are then treated as sentence equivalents (where every node in the sequence corresponds to a word) and fed to a skip-gram model, a model variant of the word embeddings introduced by

¹ Sourcecode and datasets are available at <https://doi.org/10.5281/zenodo.3514305>.

Mikolov et al. [14], which became famous under the term word2vec. node2vec [10] follows this approach, but provides means to control the random walk behavior.

For further methods and additional details of the methods mentioned above consult a survey by Goyal and Ferrara [9].

3 Additional Random Walk Modifications

In the next sections, we will introduce modifications to the random walk strategy, similar to the one implemented by node2vec [10]. The modifications can take place in two sections of the random walk algorithm: During *sampling*, we can modify the transition probabilities between nodes to draw attention to specific ones, while during *walking*, we can directly influence how the random walk traverses.

3.1 Jump Probability

We introduce the parameter j to modify the random walk during walking. It controls the probability of jumping to a random node in the graph at any given time. Intuitively, j ranges from 0 to 1, where with 0 no jumps to a random node occur, and with 1 every walking step is a random jump. The latter allows to create a truly random “walk” through the graph, without drawing any attention to the structure of the graph and edges with their respective weights. We are sampling truncated random walks, i.e., we start walks with a fixed length from every node in the graph. Therefore, the jump probability can be seen as noise in the truncated walks, opposed to jump probability in a single (huge) walk (as used by PageRank [17] for example).

3.2 Hub Attention

Hubs are nodes in a graph with a degree that greatly exceeds the average [2]. The threshold ε is the sum of the mean node degree and the standard deviation of all degrees:

$$\varepsilon = \overline{deg} + \sigma_{deg}$$

Based on that, we define the subset of hubs H of the nodes N as every node with a higher degree than ε :

$$H = \{n \in N \mid deg(n) > \varepsilon\} \subset N$$

We are now modifying the random walk during sampling. Similar to node2vec [10], we change the unnormalized transition probability π_{vx} of a node v to its neighbors x on edges (v, x) to $\pi_{vx} = \alpha_h(x) * w_{vx}$, where w_{vx} is the weight of the edge, and

$$\alpha_h(x) = \begin{cases} \frac{1}{h} & \text{if } x \in H \\ 1 & \text{else} \end{cases}$$

The parameter h introduced here controls the random walk tendency towards and away from hubs. If this parameter is set to a high value (>1), the probability of directly revisiting hubs is reduced, and the less frequented nodes are explored. On the other hand, if h is small (<1), it increases the likelihood of traversing to hubs, so the walk is more focused on the areas around hubs in the graph.

4 Evaluation

We begin this section by introducing the datasets and parameters used in our experiments, followed by the different evaluation tasks performed.

Les Misérables [1] is a network which contains the characters and their co-appearances in the novel “Les Misérables” by Victor Hugo. Every node represents a character, and an edge between two characters indicates that they appeared in the same book chapter. The graph consists of 77 nodes, connected via 254 edges. BlogCatalog [21] is a social network where every node is a blogger, and an edge between two of them represents friendship. The graph consists of 10,312 nodes, connected via 333,983 edges and assigned to one or more of 39 classes (multi-label). The classes are the topics the blogger is interested in.

We define a set of parameters for all learning algorithms to create a basis for a fair comparison. These are the embedding dimension d , the walk length l , the number of walks n and the skip-gram model [15] window size w . Furthermore the algorithm-specific parameters p, q (node2vec [10]), h (Hub Attention, c.f. Sect. 3.2) and j (Jump Probability, c.f. Sect. 3.1).

4.1 Reproduction of Les Misérables Case Study

To create the visualizations, we first learn the embeddings of the Les Misérables dataset using the respective random-walk algorithm. We then cluster these embeddings using k-means and assign the nodes of the graph colors based on their cluster. These embeddings are then visualized as a graph with Gephi [3]. For the common parameters, we used the values reported by Grover and Leskovec: embedding dimension $d = 16$, walk length $l = 80$, number of walks $n = 10$ and context window size $w = 10$.

The upper graph in the original paper [10, figure 3] reflects community structure and the values $p = 1$ and $q = 0.5$ are reported for walk parametrization. Our result in Fig. 1a also represents the community structure of the network, comparable to that in the original paper.

For the lower graph in the original paper [10, figure 3], which as per description resembles structural equivalence, $p = 1$ and $q = 2$ are specified. Even with grid-search over these and further parameters (l , n and w), no result close to the original could be produced. The graph never represented structural equivalence, but community structure as well (with 3 instead of 6 clusters), as shown in Fig. 1b.

Fig. 1. Les Misérables network with nodes colored corresponding to their cluster in embeddings created by different walk strategies. (Color figure online)

Table 1. Macro f1 scores and standard deviation ($\pm$) of the node classification task on the BlogCatalog dataset using different parameters for each algorithm.

Learner	Parameters	Score
node2vec	$p = 0.25, q = 0.25$	26.72 ± 0.72
	$p = 1, q = 1$	25.85 ± 0.59
Jump probability	$j = 1$	03.64 ± 0.14
	$j = 0.25$	25.27 ± 0.68
	$j = 0.1$	25.76 ± 0.55
Hub attention	$h = 0.5$	23.37 ± 0.59
	$h = 0.75$	25.21 ± 0.64
	$h = 4$	<u>27.44</u> ± 0.64
	$h = 8$	27.17 ± 0.52
	$h = 10$	27.39 ± 0.55

4.2 Node Classification

We are running a multi-label node classification task on the BlogCatalog dataset. Same as in the original paper [10], we use a one-vs-rest logistic regression classifier with L2 regularization. The dataset is split into training and test data, with a training fraction of 50%, and the scores are averaged over 10 random splits. Again, we use the common parameters as reported by Grover and Leskovec [10]: embedding dimension $d = 128$, walk length $l = 80$, number of walks $n = 10$ and window size $w = 10$. All results of the following sections are reported in Table 1. The underlined values are the best per walk strategy, and the bold one is the best value overall.

Reproduction of Results. We used the values $p = q = 0.25$ as reported by node2vec [10] and also included results with parameters $p = q = 1$, eliminating the influence of the parameters on the random walk, resembling a “pure” random walk like DeepWalk [18]. We were not able to reproduce the results within a single

iteration of the skip-gram model, but used five iterations instead. We suspect the difference to be due to an unrecognized change of the default hyper-parameters in gensim², the word2vec implementation used by node2vec. Our reproduced score is close to the reported one and even slightly better, which can be explained by the random factor of the experiment. From this, we can conclude that our values are consistent with those in the original paper. Furthermore, node2vec [10] performed better than non-parameterized random walks like DeepWalk, at least on this dataset and parameter settings.

Jump Probability. We set j to each of $\{1, 0.25, 0.1\}$. Expectably, the higher the jump probability is, the consistently worse the results get. However, at $j = 0.1$, i.e. 10% noise, the performance is close to DeepWalk. This indicates, that a small amount of noise in the walks does not drastically harm the performance of the resulting embeddings on the node classification task.

Hub Attention. We set h to each of $\{0.5, 0.75, 4, 8, 10\}$. As shown in Table 1, a value of $h = 4$ allows us to achieve the best value across the different strategies. When focussing on hubs ($h = 0.5, h = 0.75$), performance even drops below the score of a jump probability of 25% ($j = 0.25$), i.e. jumping to a random node in every 4th step on average. Conversely, we gain performance if we put more attention to otherwise less-frequently visited nodes (i.e. if we increase h), at least up to a certain point where the results stabilize.

5 Discussion and Conclusion

We attribute our inability to reproduce the case study in terms of structural equivalence to the skip-gram model. Its objective is to predict neighboring nodes and hence, nodes with similar neighbors are represented closeby in embedding space. Another factor is the context window size of the skip-gram model: No matter how far out a walk traverses, only nodes within this window will be considered as context. This also means, that the walks which start at a particular node are not that relevant to this particular node, but its embedding is determined by all the walks traversing through this node. With optimal parameter settings and taking the standard deviation into account, the performance difference between the walk strategies on the node classification task is negligible. In addition, Perozzi et al. report a macro-F1 score of 27.3 for DeepWalk [18], doing shorter, but more walks. They report performance to increase constantly with the number of walks until it finally stabilizes.

We conclude, that adapting the walk strategy can improve the embedding performance, if the number of sampled walks is insufficient. However, instead of tuning hyper-parameters of particular walk strategies, we can also increase the number of sampled walks per node instead. The nature of the skip-gram model

² <https://radimrehurek.com/gensim/>.

and our inability to reproduce the structural equivalence case study point to the embeddings always representing homophily.

Acknowledgments. The presented work was developed within the East-Bavarian Centre of Internet Competence, Big and Open Data Analytics for Small and Medium-sized Enterprises (BODA), funded by the Bavarian Ministry of Economic Affairs and Media, Energy and Technology.

References

1. Les misérables network dataset - KONECT, April 2017. http://konect.uni-koblenz.de/networks/moreno_lesmis
2. Barabási, A.L., et al.: Network Science. Cambridge University Press, Cambridge (2016)
3. Bastian, M., Heymann, S., Jacomy, M.: Gephi: an open source software for exploring and manipulating networks. In: Third International AAAI Conference on Weblogs and Social Media (2009)
4. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: Advances in Neural Information Processing Systems, pp. 585–591 (2002)
5. Bhagat, S., Cormode, G., Muthukrishnan, S.: Node classification in social networks. Soc. Netw. Data Anal. 115–148 (2011). https://doi.org/10.1007/978-1-4419-8462-3_5
6. Cao, S., Lu, W., Xu, Q.: Grarep: learning graph representations with global structural information. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pp. 891–900. ACM (2015)
7. Cao, S., Lu, W., Xu, Q.: Deep neural networks for learning graph representations. In: Thirtieth AAAI Conference on Artificial Intelligence (2016)
8. Ding, C.H.Q., He, X., Zha, H., Gu, M., Simon, H.D.: A min-max cut algorithm for graph partitioning and data clustering. In: Proceedings 2001 IEEE International Conference on Data Mining, pp. 107–114, November 2001. <https://doi.org/10.1109/ICDM.2001.989507>
9. Goyal, P., Ferrara, E.: Graph embedding techniques, applications, and performance: a survey. Knowl.-Based Syst. **151**, 78–94 (2018)
10. 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 (2016)
11. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks (2016). arXiv preprint [arXiv:1609.02907](https://arxiv.org/abs/1609.02907)
12. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. J. Am. Soc. Inf. Sci. Technol. **58**(7), 1019–1031 (2007). <https://doi.org/10.1002/asi.20591>
13. Maaten, L.V.D., Hinton, G.: Visualizing data using t-SNE. Mach. Learn. Res. **9**(Nov), 2579–2605 (2008)
14. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems, NIPS 2013, pp. 3111–3119. Curran Associates Inc., USA (2013). <http://dl.acm.org/citation.cfm?id=2999792.2999959>

15. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: *Advances in Neural Information Processing Systems*, pp. 3111–3119 (2013)
16. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1105–1114. ACM (2016)
17. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: bringing order to the web. Technical report Stanford InfoLab (1999)
18. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. pp. 701–710. ACM (2014)
19. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. *Science* **290**(5500), 2323–2326 (2000)
20. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1225–1234. ACM (2016)
21. Zafarani, R., Liu, H.: Social computing data repository at ASU (2009). <http://socialcomputing.asu.edu>