EasyChair Preprint

№ 5807

Task-Specific Temporal Node Embedding

Mounir Haddad, Cécile Bothorel, Philippe Lenca and Dominique Bedart

June 15, 2021

Noname manuscript No.
(will be inserted by the editor)

Task-specific Temporal Node Embedding

Mounir Haddad ·
Cécile Bothorel ·
Philippe Lenca ·
Dominique Bedart

Received: date / Accepted: date

Abstract Graph embedding aims to learn a representation of graphs' nodes in a latent low-dimensional space. The purpose is to encode the graph's structural information. While the majority of real-world networks are dynamic, literature generally focuses on static networks and overlooks evolution patterns. In a previous article entitled "TemporalNode2vec: Temporal Node Embedding in Temporal Networks", we introduced a dynamic graph embedding method that learns continuous time-aware vertex representations. In this paper, we adapt TemporalNode2vec to tackle especially the node classification-related tasks. Overall, we prove that task-specific embedding improves data efficiency significantly comparing to task-agnostic embedding.

Keywords Dynamic network embeddings · Graph representation learning · Latent representations

1 Introduction

In the last decade, literature has taken an interest in the extraction of networks' relevant structural information. Yet, one has to format graph data

M. Haddad
IMT Atlantique, Lab-STICC, F-29238 Brest, France
DSI Global Services
E-mail: mounir.haddad@imt-atlantique.fr

C. Bothorel
IMT Atlantique, Lab-STICC, F-29238 Brest, France
E-mail: cecile.bothorel@imt-atlantique.fr

P. Lenca
IMT Atlantique, Lab-STICC, F-29238 Brest, France
E-mail: philippe.lenca@imt-atlantique.fr

D. Bedart
DSI Global Services
E-mail: dominique.bedart@dsi-globalservices.fr

in such a way to make its exploitation easier for machine learning models. Traditional approaches, commonly based on user-defined heuristics [1–3], are time-consuming in terms of feature engineering.

Some emerging novel approaches known as data embeddings intend to learn a latent representation of the data that lie in graphs in low dimensional spaces [4]. These techniques have been designed for textual data at first [5] and, thereafter, have been adapted to graphs among other data structures. Last years have seen graph representation learning approaches flourish. Some are based on matrix factorization [6–8], on random walks [9–11] or on deep learning techniques (convolutional neural networks [12] or autoencoders [13,14])

As real-world datasets generally evolve over time, a few and recent approaches consider the temporal dimension to capture evolution patterns. Some methods use temporal information for the conception of more reliable embeddings [15,16], while other ones learn a representation for the network at each time step [17–19].

In a previous paper, we presented TemporalNode2vec [18], a temporal node embedding framework, based on the static node embedding algorithm Node2vec [9] and a smoothing mechanism for dynamic word embeddings [20]. Its principal advantage lies in the way temporal information is incorporated: the different time steps embeddings are learned jointly. As the majority of embedding techniques, TemporalNode2vec is task-agnostic by design: this means that the resulting vertices representations are generic and independent regarding downstream machine learning tasks. Intuitively, it seems to be possible to enhance the quality and the efficiency of the embeddings if the inference tasks to be addressed are known a priori. Upon this idea, we propose a variant of TemporalNode2vec called TsTemporalNode2vec. It is a task-specific semi-supervised embedding algorithm tied up to node classification tasks, made possible by a slight modification of the objective function of TemporalNode2vec. Our main contribution lies in the use of a part of the ground-truth nodes' labels within the representation learning process. We show in our experiments that this variant helps improving node classification tasks and makes the learned embeddings more data-efficient.

2 Task-agnostic embedding

2.1 TemporalNode2vec model

In order to build sequences of vertices, TemporalNode2vec [18] uses random walks for each time step in a similar way to node2vec [9]. Given a sequence of temporal graphs $\{G_1, \dots, G_T\}$, we obtain $T \times N \times |V|$ sequences of vertices of length l, where N is the number of walks starting from each vertex and V is the set of the temporal graph vertices.

Homophily, or the ability to bring close together embeddings which nodes share similar neighbors [21], is a property that embeddings should preserve. We

use Positive Pointwise Mutual Information matrices (PPMI) for this purpose.

$$PPMI(v_1,\, v_2)_t = max\left(0,\, log\left(\theta\, \frac{|v_1,\, v_2|_t^w \cdot |V|}{|v_1|_t \cdot |v_2|_t}\right)\right) \\ \forall (v_1,\, v_2 \neq v_1,\, t) \in V^2 \times [\![1,\, T]\!] \tag{1}$$

where $|v_i|_t$ is the number of occurrences of v_i in the set of walks of G_t, $|v_1,\, v_2|_t^w$ is the number of times v_1 and v_2 co-appear in the set of walks of G_t within a window of size w, and θ is a tunable hyperparameter controlling the trade-off between PMI entries stability (large negative values of $log\left(\theta\, \frac{|v_1,\, v_2|_t^w \times |V|}{|v_1|_t \times |v_2|_t}\right)$) and rare co-appearance pairs of nodes. The overall objective function to optimize for $\{U_1, \dots\, , U_T\}$ is:

$$\begin{aligned} L =&\; L_{St}\; + \tau\, L_{Sm}\; + \lambda\, L_{LR} \\ =& \sum_{t=1}^{T} \left\| PPMI_t - U_t\, U_t^T \right\|_F^2 + \tau \sum_{t=2}^{T} \|U_t\; -\; U_{t-1}\|_F^2\; + \lambda \sum_{t=1}^{T} \|U_t\|_F^2 \end{aligned} \tag{2}$$

where L_{St} is the static term, L_{Sm} is the temporal smoothing term and L_{LR} stands for low-rank data-fidelity enforcement.

2.2 Experiments

To evaluate TemporalNode2vec, 3 baseline methods have been considered: Two state-of-the-art static embedding methods (DeepWalk [10] and node2vec [9]) and DynamicTriad [17], a dynamic embedding approach focusing on how triads of vertices open/close. For each of those baseline methods, a grid search has been performed over multiple values for their different hyperparameters.

The performed experiments show that TemporalNode2vec improves node classification tasks (up to 14.2% in terms of F1 score) and affords to achieve good performances with a limited number of representation features (figure 1). On the other hand, TemporalNode2vec is less efficient in edge-related inference tasks.

3 Task-specific embedding

As seen above, it is possible to extract a small number of features describing accurately the input temporal graphs. This means that the dimensions of the obtained latent embedding space contain sufficient information to perform all the considered inference tasks, as the embeddings are built agnostically regarding downstream machine learning tasks. However, at this point, the distribution of each task's useful information over the latent dimensions remains unknown. For example, in a simplistic scenario, some of the retained features

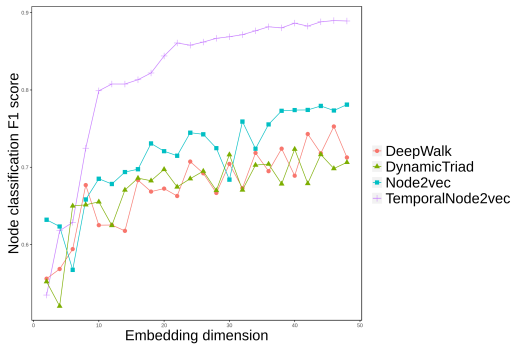

Fig. 1: Node classification on AMiner[1]dataset

may be relevant to edge reconstruction tasks and irrelevant to node classification ones. In such a configuration, to perform node classification, one can settle for a smaller number of features. Upon this idea, we propose a variant of TemporalNode2vec we call TsTemporalNode2vec (task-specific TemporalNode2vec): It is a semi-supervised embedding algorithm tied up to node classification tasks, made possible by a slight modification of TemporalNode2vec objective function.

3.1 TsTemporalNode2vec model

In addition to a sequence of graphs, TsTemporalNode2vec takes also in input a sequence of timestamped vertices labels $S = \{s_{i,c,t}\}$ where $s_{i,c,t}$ signifies that the vertex v_i belongs to the community c at the time step t. In order to conceive TsTemporalNode2vec, we modify the objective function of TemporalNode2vec by forcing the proximity of embeddings for the pairs of nodes that belong to the same community. This may be achieved by adding a term to the objective function:

$$L_{ts} = \sum_{(i,j,t)\in S_{pos}} \|u_i(t) - u_j(t)\|^2 \tag{3}$$

where $u_i(t)$ represents the embedding of the vertex v_i at t and $S_{pos} = \{(i,j,t)\}$ is derived from S and signifies that the vertices v_i and v_j belong to the same community at t. However, introducing the term L_{ts} is not practical as it involves vectors rather than matrices, unlike the other terms of the objective function shown in equation (2).

Instead, we choose to incorporate the supervision additional data into the $PPMI$ matrices: as these matrices express nodes' temporal similarities, one

[1] More details in section 3.2.2

can increase the entries corresponding to the pairs of nodes belonging to the same community. We introduce S_{sup} and $SPMI$ matrices:

$$S_{sup}(v_i,\, v_j)_t = \begin{cases} 1 \text{ if } (i,j,t) \in S_{pos} \\ 0 \text{ otherwise} \end{cases} \tag{4}$$

$$SPMI(v_i,\, v_j)_t = PPMI(v_i,\, v_j)_t + \alpha \cdot m_t \cdot S_{sup}(v_i,\, v_j)_t \tag{5}$$

where m_t is the median of the non-zero values of $PPMI_t$ and α is a hyperparameter controlling $PPMI$ values increase. We use m_t as a *normalization* so that α is of order unity.

On another note, it is relevant to notice that the timestamped labels S encompass also information about pairs of nodes belonging to different communities: in the same way as S_{pos}, we derive S_{neg} from S. It is then possible to incorporate S_{neg} in S_{sup} and $SPMI$ matrices:

$$S_{sup}(v_i,\, v_j)_t = \begin{cases} 1 & \text{if } (i,j,t) \in S_{pos} \\ -1 & \text{if } (i,j,t) \in S_{neg} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$SPMI(v_i,\, v_j)_t = max\Big(0,\, PPMI(v_i,\, v_j)_t + \alpha \cdot m_t \cdot S_{sup}(v_i,\, v_j)_t \Big) \tag{7}$$

That being set, we substitute $PPMI$ matrices with $SPMI$ ones in the objective function expressed in equation (2). The following steps of TemporalNode2vec remain unchanged.

3.2 Experiments

3.2.1 Node classification/prediction

The idea is to find nodes' ground-truth communities based on their embeddings. For the node classification task, the current vertices embeddings are used to find their labels. On the other hand, for the node class prediction task, we predict nodes' labels at a time step using their embeddings at the previous time step. For both inference tasks, the classifier used is logistic regression.

3.2.2 Datasets

In order to challenge TsTemporalNode2vec performances, we submit its output embeddings to node classification for the 3 real-world datasets.

– **AMiner** [22]: 51k researchers and 624k coauthor relationships, divided into 17 timestamped weighted graphs, (a weight represents the number of common papers between 2 nodes). Authors are mapped to research fields (labels) regarding the domains their publications address.

- **Yelp**[2]: an extract of Yelp challenge dataset. It traces internet users' comments on businesses. We consider users and businesses as being nodes while comments are regarded as interactions (38k nodes, 744k edges, 17 time steps). Users can be mapped to categories (labels) when looking at the type of businesses they usually comment on.
- **Tmall**[3]: an extract of the sales at Tmall.com 6 months before the "Double 11 Day" event in 2014. It stores buyers' interactions with products (27k nodes, 2.9M edges, 10 time steps). We assign labels to users as described for Yelp dataset (categories of preferred products as labels).

3.2.3 Labeling ratio

As TsTemporalNode2vec is a semi-supervised embedding model, one has to define and set the labeling ratio. For the datasets we consider, it is important to note that the nodes' labels are not equal regarding the *amount of information* they provide: some nodes interact within all the time steps while others are active during very few ones. Therefore, it seems worthwhile to rate nodes' contributions in terms of the provided information. For each node v_i, we define the quantity Q_i:

$$Q_i = \frac{q_i}{\sum\limits_{v_j \in V} q_j} \tag{8}$$

where q_i is the number of time steps where the vertex v_i is active. Then, a labeling ratio r can be obtained by a choosing a random sample of nodes RS_r satisfying:

$$\sum_{v_j \,\in\, RS_r} Q_j \;\approx\; r \tag{9}$$

3.2.4 Model tested values

Besides TemporalNode2vec hyperparameters, TsTemporalNode2vec has an additional one α, as described in equations (5) and (7). For our experiments, we keep the values of TemporalNode2vec hyperparameters giving the best performances on node classification and perform a grid search over $(d,\ r,\ \alpha) \in \{2,\ 4,\ 7,\ 10,\ 15,\ 20,\ 25\} \times \{0.1,\ 0.25,\ 0.5\} \times \{0.1,\ 0.33,\ 0.67,\ 1\}$.

3.2.5 Results analysis and interpretation

Table 1 shows the results of the experiments. Overall, TsTemporalNode2vec improves the performances on node classification tasks. This is particularly the case for small embedding dimensions. This finding confirms our prior intuition stating that task-specific embedding captures as much relevant information as possible within the embedding latent dimensions. Also, as the embedding

[2] https://www.yelp.com/dataset/challenge
[3] https://tianchi.aliyun.com/competition/entrance/231576/information

dimension grows, the gap between task-specific and task-agnostic approaches scores tends to narrow: for large values of d, the relevant information is captured by both methods.

Concerning the labeling ratio, the experiments show that setting r to 0.1 is sufficient to enhance the F1 score. This means that labeling 10% of the nodes helps to significantly improve the inference performances. On the other hand, it seems that larger values of r (0.25 and 0.5) do not bring much more improvement to the performances (respectively, an average F1 score gain of 0.8% and 1.4% comparing to $r = 0.1$). This could be explained by the fact that TsTemporalNode2vec needs a small ratio of labeled nodes to understand the kind of communities to focus on.

Lastly, the hyperparameter α seems to have a significant impact on the performances. For example, large α values result in bad performances for AMiner dataset, while setting α to 0.33 would be the best choice for Yelp dataset.

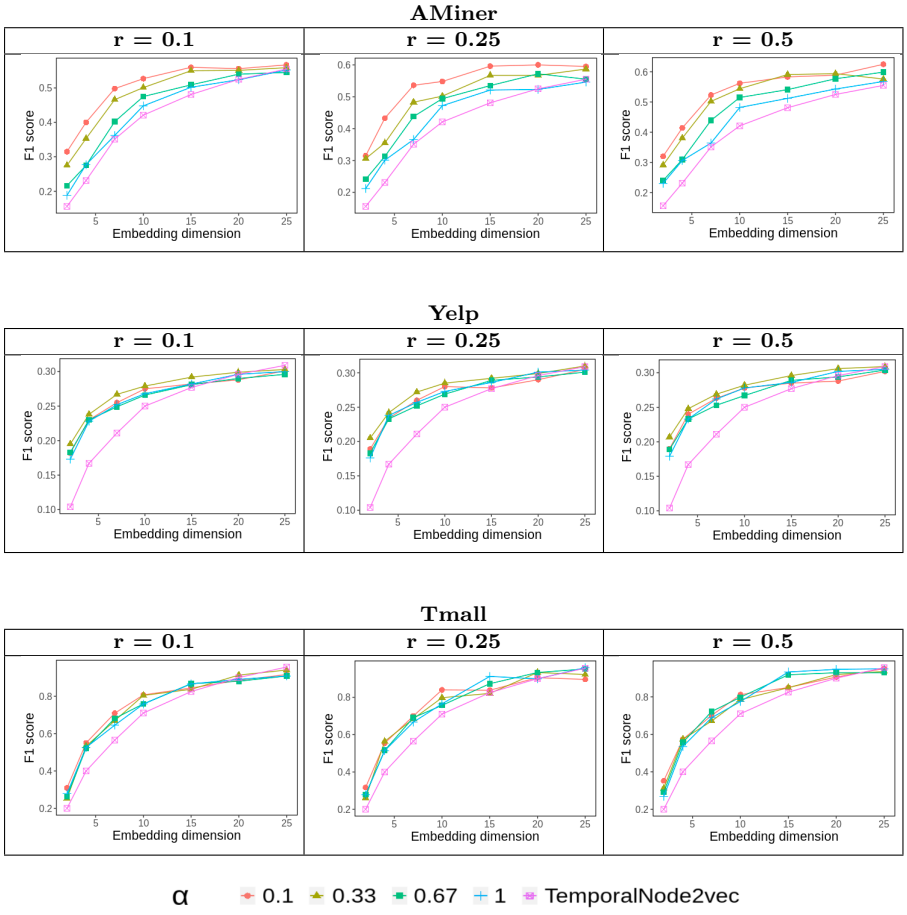

Table 1: TsTemporalNode2vec results analysis

Concerning Tmall dataset, it seems that the value of α does not have a huge impact on the classification score. Further investigation is needed to explain that point. Overall, it seems that the optimal value of α and its impact highly depends on the considered dataset.

3.3 TsTemporalNode2vec application context

Task-specific temporal graph embedding can be useful in many cases. One can take advantage of total or partial metadata giving information about nodes communities. Also, when such data is not available, it is possible to perform a *soft* labeling step, consisting of marking pairs of nodes probably belonging to the same community or probably belonging to different communities, based on some user-defined heuristic. In this case, task-specific embedding offers more flexibility, as it is possible to specify and highlight the desired communities or types of communities. One must nevertheless note that the hand-engineered co-belonging rule is to be considered as a method to reproduce the ground-truth communities partially but reliably. It should then privilege precision rather than recall.

4 Conclusion

In this paper, we presented TsTemporalNode2vec, a task-specific temporal node embedding method, tied up to node classification. We proved its efficiency in encoding dynamic graphs structural information into a very limited number of dimensions. Moreover, we listed some application contexts where task-specific embedding could be useful. Further work related to exploiting the embeddings for different purposes (visualization, dynamics analysis and prediction) is underway.

References

1. S. Bhagat, G. Cormode, S. Muthukrishnan, in *Social network data analytics* (Springer, 2011), pp. 115–148
2. S.V.N. Vishwanathan, N.N. Schraudolph, R. Kondor, K.M. Borgwardt, Journal of Machine Learning Research **11**(Apr), 1201 (2010)
3. D. Liben-Nowell, J. Kleinberg, Journal of the American society for information science and technology **58**(7), 1019 (2007)
4. Y. Bengio, A. Courville, P. Vincent, IEEE transactions on pattern analysis and machine intelligence **35**(8), 1798 (2013)
5. T. Mikolov, K. Chen, G. Corrado, J. Dean, arXiv preprint arXiv:1301.3781 (2013)
6. S. Cao, W. Lu, Q. Xu, in *Proceedings of the 24th ACM international on conference on information and knowledge management* (ACM, 2015), pp. 891–900
7. M. Belkin, P. Niyogi, in *Advances in neural information processing systems* (2002), pp. 585–591
8. M. Ou, P. Cui, J. Pei, Z. Zhang, W. Zhu, in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (ACM, 2016), pp. 1105–1114

9. A. Grover, J. Leskovec, in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (ACM, 2016), pp. 855–864
10. B. Perozzi, R. Al-Rfou, S. Skiena, in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (ACM, 2014), pp. 701–710
11. H. Chen, B. Perozzi, Y. Hu, S. Skiena, in *Thirty-Second AAAI Conference on Artificial Intelligence* (2018)
12. T.N. Kipf, M. Welling, arXiv preprint arXiv:1609.02907 (2016)
13. S. Cao, W. Lu, Q. Xu, in *Thirtieth AAAI Conference on Artificial Intelligence* (2016)
14. D. Wang, P. Cui, W. Zhu, in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (ACM, 2016), pp. 1225–1234
15. Y. Zuo, G. Liu, H. Lin, J. Guo, X. Hu, J. Wu, in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (ACM, 2018), pp. 2857–2866
16. G.H. Nguyen, J.B. Lee, R.A. Rossi, N.K. Ahmed, E. Koh, S. Kim, in *Companion of the The Web Conference 2018 on The Web Conference 2018* (International World Wide Web Conferences Steering Committee, 2018), pp. 969–976
17. L. Zhou, Y. Yang, X. Ren, F. Wu, Y. Zhuang, in *Thirty-Second AAAI Conference on Artificial Intelligence* (2018)
18. M. Haddad, C. Bothorel, P. Lenca, D. Bedart, in *International Conference on Complex Networks and Their Applications* (Springer, 2019), pp. 891–902
19. S. Mahdavi, S. Khoshraftar, A. An, in *2018 IEEE International Conference on Big Data (Big Data)* (IEEE, 2018), pp. 3762–3765
20. Z. Yao, Y. Sun, W. Ding, N. Rao, H. Xiong, in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (ACM, 2018), pp. 673–681
21. S. Fortunato, Physics reports **486**(3-5), 75 (2010)
22. J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, Z. Su, in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (ACM, 2008), pp. 990–998