A Study of Dynamic Heterogeneous Network Prediction based on DyHATR-Skip Embedding Fusion

Zhaoke Li[1], Shuwei Xu[2*], Gaofei Si[2] and Jingyun Zhang[2]

[1]Henan University School of Software, Kaifeng, Henan, China
[2]Henan University School of Software, Kaifeng, Henan, China

ke@henu.edu.cn;
*Corresponding author: xsw@henu.edu.cn;
1095738316@qq.com; zjingy599@163.com

Abstract. To solve the problem that the single dynamic heterogeneous network embedding method (DyHATR) cannot capture the node features accurately and adequately, which leads to the low efficiency of the final link prediction. This paper proposes to solve this problem by using the DyHATR based on the Skip-gram method (DyHATR-Skip): (1) Generating word embedding by using the Skip-gram model in Word2vec; (2) Fusing the generated word embedding with the node embedding generated by DyHATR for splicing fusion, which is named as DyHATR-Skip. The method generates new node embedding by DyHATR and Skip-gram models. The experimental results show that the DyHATR-Skip method proposed in this paper performs better than the single DyHATR method. In the DyHATR-Skip method, AUROC improves 0.07, 0.01, 0.05 and AUPRC improves 0.07, 0.01, 0.03 on Twitter, Math-Overflow and EComm datasets respectively. Therefore, the DyHATR-Skip method proposed in this paper can capture node features and generate node embedding more fully and accurately compared to single network embedding methods and has better performance in dynamic link prediction. But since words and vectors are one-to-one in Word2vec, DyHATR-Skip has some limitations for multisense words and complex datasets.

Keywords: Dynamic heterogeneous networks; DyHATR model; Skip-gram model; Embedding fusion

1 Introduction

Existing network embedding methods have made some progress in link prediction. Still, they usually deal with static networks (nodes do not change over time) or homogeneous networks (there is only one type of node-node relationship or node-node interaction in the network)[1]. However, nodes and edges in real-world networks are usually heterogeneous and dynamic. The nodes and edges are of multiple types, and the network is constantly being updated as time changes, with different types of nodes and edges being added to the network at any time. The network is constantly evolving[2].

2 Related Work

2.1 Development of network embedding

The main existing network embedding methods are static homogeneous networks [3,4,5], static heterogeneous networks [6,7,8,9], and dynamic homogeneous networks [10,11,12,13]. Dynamic network embedding is usually described as a static network snapshot of an ordered list [14,15], divided into different snapshots based on a specific time, such as one day as a snapshot in the literature. Over time, the dynamic network can predict the next linked node in the snapshot by capturing the evolutionary patterns of different snapshots.

At present, dynamic heterogeneous network embedding is less studied. In 2021, Xue H proposed the latest dynamic heterogeneous network embedding method (DyHATR), but the process cannot accurately capture node features.

2.2 DyHATR Model

The DyHATR model proposed by Xue H is a dynamic heterogeneous network embedding method, which uses a hierarchical attention model to acquire the heterogeneity of nodes and edges in the network and uses a temporal attention neural network model to obtain the evolutionary patterns of snapshots in the network. The specific framework of the whole network is shown in Figure 1 below[1].

3 Embedding Fusion

3.1 DyHATR-Skip method

Through analyzing and studying DyHATR, HErec, and the Skip-gram model in Word2vec, proposes an embedding fusion method called DyHATR-Skip through a series of experiments. DyHATR-Skip distinguishes node and edge heterogeneity by introducing a hierarchical attention model divided into node-level attention and edge-level attention. Node-level attention is used to learn the neighbor weight of each node, and then the important features of these nodes are aggregated into a new node representation:

$$\alpha_{i,j}^{rt} = \frac{exp(\sigma(a_r^T \cdot [W^r \cdot x_i || W^r \cdot x_j]))}{\sum_{k \epsilon N_i^{rt}} exp(\sigma(a_r^T \cdot [W^r \cdot x_i || W^r \cdot x_k]))} \tag{1}$$

$$f_i^{rt} = \sigma\left(\sum_{j \epsilon N_i^{rt}} \alpha_{ij}^{rt} \cdot W^r \cdot x_j\right) \tag{2}$$

$$h_i^{rt} = concat(f^1, f^2, ..., f^k) \tag{3}$$

In Equation (1), $\alpha_{i,j}^{rt}$ represents the weight coefficient of the node pair (i, j) with edge type r in the t-th snapshot. In Equation (2), f_i^{rt} represents the final representation of node i with edge type r in the t-th snapshot. In Equation (3), f^k is the f_i^{rt} shorthand,

and k is the number of heads of multi-headed attention, which $\mathrm{h}_{\mathrm{i}}^{\mathrm{rt}}$ indicates the multi-headed attention representation of node i with edge type r in the t-th snapshot.

To achieve and aggregate specific information about each node's different edge types and generate the final node representation, DyHATR proposes edge-level attention with the following Equation.

$$\beta_{\mathrm{i}}^{\mathrm{rt}} = \frac{\exp(\mathrm{q}^{\mathrm{T}} \cdot \sigma(\mathrm{W} \cdot \mathrm{h}_{\mathrm{i}}^{\mathrm{rt}} + \mathrm{b}))}{\sum_{r \in R} \exp(\mathrm{q}^{\mathrm{T}} \cdot \sigma(\mathrm{W} \cdot \mathrm{h}_{\mathrm{i}}^{\mathrm{rt}} + \mathrm{b}))} \quad h_i^t = \sum_{r=1}^{R} \beta_{\mathrm{i}}^{\mathrm{rt}} \cdot \mathrm{h}_{\mathrm{i}}^{\mathrm{rt}} \tag{4}$$

In Equation (4), the previous three formulas are summarized, and the specific edges are embedded in the set to acquire the final representation of node i in the t-th snapshot.

DyHATR-Skip uses a time-level self-attentive model to further capture the evolutionary patterns on the dynamic network, rather than splicing all feature vectors together as a final embedding to predict dynamic links. The temporal attention model can be defined as:

$$Z_i = \Gamma_i - V_i = softmax\left(\frac{(S_i W_q)(S_i W_k)^T}{\sqrt{D'}} + M\right) - (S_i W_v) \tag{5}$$

where $\Gamma_i \in R^{T \times T}$ is the importance matrix; $M \in R^{T \times T}$ denotes the mask matrix.

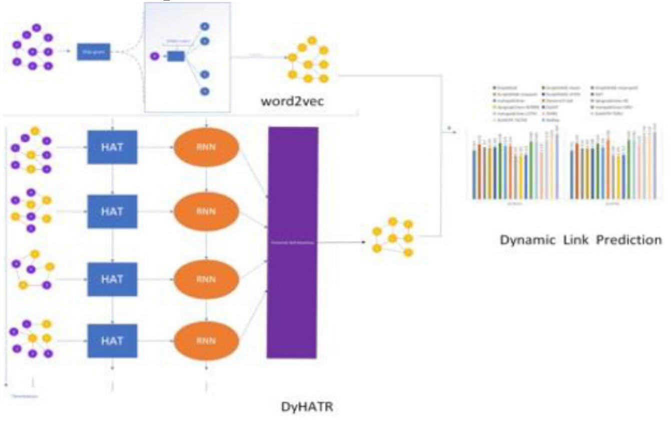

Fig. 1. Model structure of DyHATR-Skip

3.2 Loss function

DyHATR-Skip uses the embedding fusion method to fuse DyHATR and Skip-gram, so the loss function is set as the sum of the two. The loss function formula of DyHATR-Skip is defined as:

$$\mathrm{L} = \mathrm{L}(z_u^T) + \tau = \sum_{v \in N^T(u)} -\log\left(\sigma(< z_u^T, z_v^T >)\right) - Q \cdot E_{v_n \sim p_{n(v)}} \log\left(\sigma(< -z_u^T, z_v^T >)\right) + \lambda \cdot L_p + \sum_{w \epsilon C} \sum_{u \epsilon content(w)} \sum_{z \epsilon \{u\} \cup NEG(u)} L^u(z) \cdot \log[\sigma(v(w)^T \theta^z)] + \left(1 - L^u(z)\right) \cdot \log[1 - \sigma(v(w)^T \theta^z)] \tag{6}$$

Where σ is the activation function (such as the sigmoid function); <> is the inner product operation. $N^T(u)$ is the last t snapshots when node u has a fixed length of randomly walking neighbors; P_n (v) denotes the negative sampling distribution; Q denotes the number of negative samples; L_p is the penalty term of the loss function to avoid overfitting, such as L_2 regularization; λ is the hyperparameter controlling the penalty function; C is the dataset; Context(w) denotes the context of word w; NEG(u) denotes the negative sample subset of u; $L^u(z)$ is the label of the word u, which is 1 when u = z and 0 otherwise; v(w) denotes the word vector of word w; θ^z indicates an auxiliary vector corresponding to word z, which is the parameter to be trained.

4 Experiment

4.1 Experimental environment and datasets

The experiment in this paper was carried out in three real datasets under the Linux server Ubuntu 18.04. The experimental environment is a server with a GPU of Nvidia Geforce RTX 2080 and a CPU of 6× Xeon E5-2678 v3. The experimental platform is a professional version of PyCharm with Python version 3.6, Tensorflow 2.2, CUDA 10.1, and Cudn 7.6.5. The information on datasets is shown in Table 1.

Table 1. Information on the three datasets

Datasets	Nodes	Edges	Node Types	Edge Types	Snapshots
Twitter	100000	63410	1	3	7
Math-Overflow	24818	506550	1	3	11
EComm	37724	91033	2	4	11

4.2 Analysis of experimental results

The experimental results of DyHATR-Skip and DeepWalk, GraphSAGE, GAT, and other methods on three real datasets are shown in Figure 2. From Figure 2, it can be seen that DyHATR-Skip achieves the best AUROC and AUPRC on the three different datasets.DyHATR-Skip achieves the highest AUROC and AUPRC on the Math-Overflow dataset with 0.7638 and 0.8060, respectively. For Twitter, DyHATR-Skip has a 7% improvement in AUROC over the second highest DyHATR-TLSTM (0.660); for EComm, DyHATR-Skip also has a better performance on AUROC than DyHATR-TLSTM (0.696) by nearly 5%, again having better performance than other algorithms.

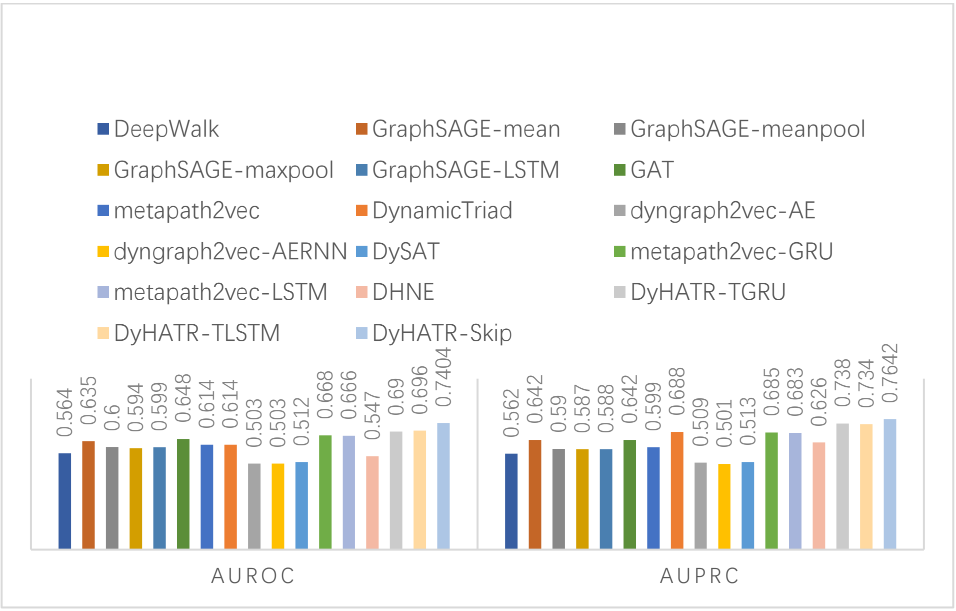

Fig. 2. Experimental results of dynamic linking of individual models on the EComm dataset

5 Conclusion

In recent years, network embedding methods have made significant progress and have been widely used in various fields. However, most of the existing network embedding methods are aimed at static or homogeneous networks and rarely can handle dynamic heterogeneous networks until DyHATR, which can handle both dynamic and heterogeneous. But, the drawback of DyHATR is the low AUROC metric, while DyHATR-Skip improves the AUROC and AUPRC of DyHATR by fusing DyHATR and Skip-gram. It is demonstrated experimentally with three real datasets that the DyHATR-Skip method achieves a significant improvement compared to DyHATR.

Acknowledgments

This study was funded by Engineering Ethics, a quality course of Henan Province Post-graduate Quality Improvement Project in 2022 (Project Approval No.: YJS2022KC29).

References

1. Xue H, Yang L, Jiang W, et al. Modeling dynamic heterogeneous network for link prediction using hierarchical attention with temporal rnn [J]. arXiv preprint arXiv:2004.01024, 2020.

2. Shi C, Hu B, Zhao W X, et al. Heterogeneous information network embedding for recommendation [J]. IEEE Transactions on Knowledge and Data Engineering, 2018, 31(2): 357-370.
3. Perozzi, B., Al-Rfou, R., & Skiena, S. (2014, August). Deep walk: Online learning of social representations. in Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 701-710).
4. Grover, A., & Leskovec, J. (2016, August). node2vec: Scalable feature learning for networks. in Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 855-864).
5. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015, May). Line: Large-scale information network embedding. in Proceedings of the 24th international conference on world wide web (pp. 1067-1077).
6. Dong, Y., Chawla, N. V., & Swami, A. (2017, August). metapath2vec: Scalable representation learning for heterogeneous networks. in Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 135-144).
7. Zhang, H., Qiu, L., Yi, L., & Song, Y. (2018, July). Scalable Multiplex Network Embedding. in IJCAI (Vol. 18, pp. 3082-3088).
8. Xue, H., Peng, J., Li, J., & Shang, X. (2019, November). Integrating multi-network topology via deep semi-supervised node embedding. in Proceedings of the 28th ACM International Conference on Information and Knowledge Management (pp. 2117-2120).
9. Cen, Y., Zou, X., Zhang, J., Yang, H., Zhou, J., & Tang, J. (2019, July). Representation learning for attributed multiplex heterogeneous network. in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 1358-1368).
10. Goyal, P., Kamra, N., He, X., & Liu, Y. (2018). Dynam: Deep embedding method for dynamic graphs. arXiv preprint arXiv:1805.11273.
11. Goyal, P., Chhetri, S. R., & Canedo, A. (2020). dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. Knowledge-Based Systems, 187, 104816.
12. Zhou, L., Yang, Y., Ren, X., Wu, F., & Zhuang, Y. (2018, April). Dynamic network embedding by modeling triadic closure process. in Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 32, No. 1).
13. Sankar, A., Wu, Y., Gou, L., Zhang, W., & Yang, H. (2018). Dynamic graph representation learning via self-attention networks. arXiv preprint arXiv:1812.09430.
14. Singer, U., Guy, I., & Radinsky, K. (2019). Node embedding over temporal graphs. arXiv preprint arXiv:1903.08889.
15. Chen, J., Zhang, J., Xu, X., Fu, C., Zhang, D., Zhang, Q., & Xuan, Q. (2019). E-lstm-d: A deep learning framework for dynamic network link prediction. IEEE Transactions on Systems, Man, and Cybernetics: Systems.