Influencer Detection with Dynamic Graph Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Leveraging network information for prediction tasks has become a common practice in many domains. Being an important part of targeted marketing, influencer detection can potentially benefit from incorporating dynamic network representation. In this work, we contribute to the literature by investigating different dynamic Graph Neural Networks (GNNs) configurations for influencer detection and evaluate their prediction performance using a unique corporate data set. We show that encoding temporal attributes and having a history of node dynamics prior to making predictions significantly impact performance. Furthermore, our empirical evaluation illustrates that network centrality measures are more beneficial than capturing neighborhood representation.

1 Introduction

Advances in data collection and processing have enhanced the use of automated data workflows for decision-making. A primary source of data comes in the form of networks that capture connections between people. When relational information is leveraged, it is assumed that people in the network influence each other's behavior and decisions, which has been shown to be true in many domains such as fraud detection [Baesens et al., 2015] or e-commerce recommendations [Sun et al., 2015].

A common way information flows through a network is by the Word of Mouth effect which is seen as a powerful tool for spreading influence among customers in marketing [Puigbo et al., 2014]. Customers who succeed in utilizing this effect in order to change others perspectives are considered influencers [Rogers and Cartano, 1962]. It is possible to model such scenarios as a network, in which its topology plays a crucial role in identifying influencers. This is a large area of study with many standard approaches for encoding the network topology, such as neighborhood and centrality metrics as well as collective inference algorithms Baesens et al. [2015]. Due to the rising popularity of deep learning, graph neural networks (GNNs) are extensively used for end-to-end tasks of graph learning [Rhee et al., 2017, Guo et al., 2019]. However, the research on influencer detection with GNNs is limited especially when it comes to networks that evolve in time and when there exist several types of connections in a network. To the best of our knowledge, there exist no research about influencer detection on dynamic attributed edge-colored networks with GNNs. The main purpose of this paper is to add to the body of research on influencer detection by evaluating different dynamic GNNs configurations and investigating whether encoding network topology with GNNs and capturing the dynamic evolution of the network have an added value to performance. By doing so, the following contributions are made. Firstly, we adapt dynamic GNNs for ex-post influencer detection, that is, identifying users of the product who influence neighboring individuals to acquire it in the future. Secondly, we evaluate different GNN configurations in combination with different

Submitted to NeurIPS 2022 Temporal Graph Learning Workshop. Do not distribute.

RNN configurations for our problem[1]. Finally, we compare the results to baseline static graph neural networks and dynamic non-GNN approaches.

2 Related work

Puigbo et al. [2014] highlight the importance of influencer detection since the rise of the Internet. However, most of its traditional approaches lack more advanced indicators of the relationships between network actors. Due to the rising popularity of GNNs and the demand for improved relationship extraction techniques, the graph influence network framework has been proposed by Shi et al. [2022]. The framework is aimed at finding the influential neighbors of a node. However, it is not designed for detecting global influencers and can be applied on static graphs only.

Networks can be seen as an unstructured data source, requiring specific methods in order to extract network topology and be able to incorporate it into prediction models. Some approaches to learning on graphs are based on matrix factorization including spectral clustering [Von Luxburg, 2007] and learning with modularity matrix [Tang and Liu, 2009]. More advanced methods are based on learning by performing random walks on a graph, e.g., DeepWalk [Perozzi et al., 2014] and node2vec [Grover and Leskovec, 2016]. The enhancements in deep learning have brought GNNs into the forefront of the field where they demonstrate cutting-edge performance [Zhou et al., 2020]. The general design pipeline of GNNs includes the steps of specifying the graph type and scale, deciding on the task type and building the model using carefully designed computational modules [Zhou et al., 2020].

3 Influencer detection with Discrete Time Dynamic Graphs

Following the design pipeline of Zhou et al. [2020], we define the task of future influencer detection in this paper as a supervised node-level learning problem on a dynamic heterogeneous undirected graph. A typical graph learning process consists of an encoder and decoder [Hamilton et al., 2017]. The encoder part of the model is aimed at learning node embeddings while the decoder part is used to solve a prediction task, e.g., node classification. Zhu et al. [2022] survey different encoder-decoder architectures that exist for supervised dynamic graph learning and classify them into Discrete Time Dynamic Graph (DTDG) learning that uses network snapshots and Continuous Time Dynamic Graph (CTDG) learning that deals with an updating event stream in a network. Following their taxonomy, we capture the graph topology at each timestamp by applying the DTDG encoder for attributed static graphs, namely, Graph Convolutional Networks (GCNs) [Kipf and Welling, 2017] and Graph Attention Networks (GATs) [Veličković et al., 2018]. We capture the dynamic nature of the networks by employing the models from the RNN family as a DTDG decoder. We also compare these models with baseline models, namely, the PageRank+RNN, static GNNs and dynamic non-GNN models.

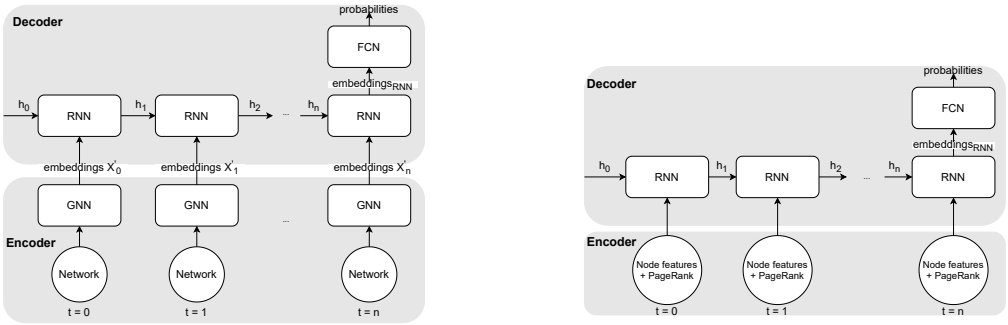
3.1 GNN + RNN

In order to deal with complex data structures and arbitrary size of neighbors, Kipf and Welling [2017] propose Graph Convolutional Networks that can learn on non-euclidean data such as graphs. A GCN model learns node embeddings by aggregating information from a node's neighborhood. However, it assigns the same importance to all the neighboring nodes which is rarely the case in practice. Hence, Veličković et al. [2018] introduce Graph Attention Networks where nodes follow a self-attention mechanism and assign an importance to each connection by attending over their neighbors.

The disadvantage of both GCN and GAT architectures is that they are static and do not take into account the dynamic changes in the network that happen over time. To take into account the time dimension, we use the GNN models mentioned above together with Recurrent Neural Networks, namely the Long Short-Term Memory (LSTM) model [Hochreiter and Schmidhuber, 1997] and Gated Recurrent Units (GRUs) model [Cho et al., 2014]. The LSTM model is capable of learning long-term dependencies by storing long-term memory in a cell state while capturing the most recent information in its hidden states. The GRU model is a less complex version of LSTMs as it does not have a cell state and stores long-term memory directly in its hidden states. In both the LSTM and GRU models, the output embeddings of a GNN model are used as an input.

[1]The link to the source code is censored to preserve anonymity

Considering all the above, we investigate four different model configurations, i.e., GCN+LSTM, GCN+GRU, GAT+LSTM, and GAT+GRU. Their architecture is displayed in Figure 1a.



(a) GNN+RNN

(b) PageRank+RNN

Figure 1: Models' architecture

3.2 PageRank + RNN & Features+RNN

Following the strategy used in the research by Óskarsdóttir et al. [2017], we can enrich non-relational classifiers with network features. One of such network features that can summarize node importance is PageRank [Brin and Page, 1998]. As the PageRank value represents the relative importance of the node within one component, we calculate the value for PageRank separately within each of the connected components. PageRank values are used as an additional node feature and subsequently utilized for dynamic node classification with LSTMs and GRUs (Figure 1b).

4 Experiments

We utilize data from one city of a Super-App company operating in Latin America that has both a delivery app and issues credit cards to its customers. We construct monthly network snapshots based on different types of connections between the users. The resulting network can be categorised as a dynamic attributed undirected network with implicit time and colored edges. The nodes in the graph represent the customers and are attributed with the features that characterize the customer's credit card usage at each time snapshot. The edges in the graph are colored by different types of connections between network actors as displayed in Appendix A, Table A.1. Coloring the edges is performed by creating four binary edge features (transformed to edge weights in the GCN+RNN models). We note that existing connections never disappear from the network while it is possible that new connections are established. The graph is labelled: customers who referred (i.e. extended an invitation to the Super-App's services to someone they are connected to) other customers in the past at least once are labelled as influencers while other customers are labelled as non-influencers.

The network is imbalanced as being an influencer is less common than being a non-influencer (imbalance ratio is $\sim 12\%$). We check if oversampling helps to handle the data imbalance during the model validation step and apply the Synthetic Minority Oversampling TEchnique (SMOTE) [Chawla et al., 2002] for the embeddings generated by the RNN model. We follow the oversampling strategy of Zhao et al. [2021] and oversample the nodes with the embeddings generated by the RNN model or by the GNN model where the RNN model is not utilized. We generate synthetic nodes of the minority class of different quantities that we set as a hyperparameter (Appendix A, Table A.2).

The data splits, a general pipeline of training, validating and testing the models as well as the best hyperparameter specifications found by grid search are displayed in Appendix A, Figure A.1 and Table A.2, respectively. The type of resources used is displayed in Appendix A, Table A.3.

5 Results and Discussion

As can be seen from Table 1, GCN+RNN models obtain the highest AUC values on validation set with seen nodes while obtaining lower AUC values on validation and test sets with unseen nodes. Since seen nodes were in the training network, the models can generalize for them while the dynamics of unseen nodes are not clear and the models cannot learn their representation well.

Table 1: Models performance (models are implemented using pytorch geometric library [Fey and Lenssen, 2019]). Test AUC is calculated for the best model according to validation AUC. Confidence intervals of test (val.) AUC values are obtained from bootstrapping test (val.) AUC values.

Model	Val. AUC seen nodes	Val. AUC unseen nodes	Test AUC seen nodes	Test AUC unseen nodes	Total time, seconds
GCN+LSTM	**0.721±0.018**	0.578±0.005	0.666±0.017	0.573±0.007	11740.9
GCN+GRU	**0.721±0.017**	0.577±0.005	0.643±0.016	0.576±0.007	3481.5
GAT+LSTM	0.695±0.017	0.584±0.005	0.658±0.016	0.582±0.007	14266.5
GAT+GRU	0.700±0.017	0.570±0.005	0.645±0.017	0.567±0.007	3909.3
PageRank+LSTM	0.711±0.014	0.643±0.002	0.681±0.011	0.637±0.007	1746.0
PageRank+GRU	0.718±0.014	0.652±0.003	**0.685±0.012**	0.646±0.007	882.7
Features+LSTM	0.710±0.014	0.644±0.002	0.679±0.011	0.639±0.007	2054.9
Features+GRU	0.716±0.014	**0.661±0.002**	0.679±0.010	**0.650±0.007**	1662.9
Static GAT	0.587±0.018	0.525±0.004	0.581±0.019	0.524±0.008	4388.8
Static GCN	0.691±0.018	0.543±0.004	0.653±0.030	0.534±0.006	1853.3

In contrast, non-GNN models, namely, PageRank+RNN and Features+RNN models perform consistently better on unseen nodes. For seen nodes, the model enriched with PageRank values obtains a slightly better performance with GRU decoder while it is not the case for unseen nodes. Moreover, for these models, the GRU decoder performs better than the LSTM decoder while also being faster. Overall, non-GNN models can be seen as a clear winner for the predicting influencers on unseen nodes while there is no evident dominance of any of the models on unseen nodes.

Dynamic GNN and non-GNN models perform better than non-dynamic GNN models; hence, capturing time-evolving patterns play an important role in influencer detection. What is more, the static GAT model performs worse than its GCN counterpart meaning that computationally expensive attention does not have an added value to the performance; a similar pattern occurs for GNN-RNN models where the models with and without attention perform similarly.

We note that upsampling does not increase the performance meaning that the models can deal with the data imbalance (Appendix A, Table A.2). Also, most of the GNN+RNN models are shallow with only two GNN layers; hence, expensive addition of layers does not results in an enhanced performance.

6 Conclusion

Early detection and targeting of influencers allows for efficient spread of information through the network. Hence, different model architectures for influencer detection with networks should be evaluated. For these reasons, we researched different dynamic GNNs configurations and investigated whether encoding network topology with GNNs and capturing the dynamic evolution of the network have an added value to the prediction performance.

First, we conclude that dynamics of the network play an important role in the influencer detection; thus, the decoder of the model should capture time. Second, centrality measures such as PageRank are more important than neighbor feature representations captured by GNNs. Third, it is better to perform influencer prediction on unseen nodes with non-GNN dynamic models while more analysis is required to investigate the best model for seen nodes. As the use of the influencer detection model is intended for marketing, we foresee the best models will allow optimizing the frequency and tenor of targeted marketing actions some users can be subjected to.

Our work has several limitations. First, we use only credit card features that could limit representation power. Another limitation is that there could exist other connections in the network that are not captured by a current network setup. Moreover, the way edges are created based on the Geohash 7 proximity can affect the connectivity strength of the network. Future improvements include unsupervised learning algorithms for influencer detection. Next, we can evaluate models from a profit-driven perspective that will enable us to bring more business context into the results. Third, we can explore behavioural node features from the delivery app which can potentially increase predictive power. Moreover, we can expand the network to more than one city to make the results more generalizable.

References

B. Baesens, V. Van Vlasselaer, and W. Verbeke. *Fraud analytics using descriptive, predictive, and social network techniques: a guide to data science for fraud detection*. John Wiley & Sons, 2015.

S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

M. Fey and J. E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 922–929, 2019.

W. L. Hamilton, R. Ying, and J. Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=SJU4ayYgl`.

M. Óskarsdóttir, C. Bravo, W. Verbeke, C. Sarraute, B. Baesens, and J. Vanthienen. Social network analytics for churn prediction in telco: Model building, evaluation and network architecture. *Expert Systems with Applications*, 85:204–220, 2017.

B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

J.-Y. Puigbo, G. Sánchez-Hernández, M. Casabayó, and N. Agell. Influencer detection approaches in social networks: A current state-of-the-art. In *CCIA*, pages 261–264, 2014.

S. Rhee, S. Seo, and S. Kim. Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification. *arXiv preprint arXiv:1711.05859*, 2017.

E. M. Rogers and D. G. Cartano. Methods of measuring opinion leadership. *Public opinion quarterly*, pages 435–441, 1962.

Y. Shi, P. Quan, Y. Xiao, M. Lei, and L. Niu. Graph influence network. *IEEE Transactions on Cybernetics*, 2022.

Z. Sun, L. Han, W. Huang, X. Wang, X. Zeng, M. Wang, and H. Yan. Recommender systems based on social networks. *Journal of Systems and Software*, 99:109–119, 2015. ISSN 0164-1212. doi: https://doi.org/10.1016/j.jss.2014.09.019. URL `https://www.sciencedirect.com/science/article/pii/S0164121214002064`.

L. Tang and H. Liu. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 817–826, 2009.

P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`.

U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

T. Zhao, X. Zhang, and S. Wang. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 833–841, 2021.

J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

Y. Zhu, F. Lyu, C. Hu, X. Chen, and X. Liu. Encoder-decoder architecture for supervised dynamic graph learning: A survey, 2022. URL `https://arxiv.org/abs/2203.10480`.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Section 1.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Table 1
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix A, Table A.3.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Table 1.
   (b) Did you mention the license of the assets? [Yes] See Table 1.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See Section 1.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] Users consent to share the data by agreeing to ToS of the company. The ethics approval for this project has been passed.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] The data have been fully anonymized.
5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

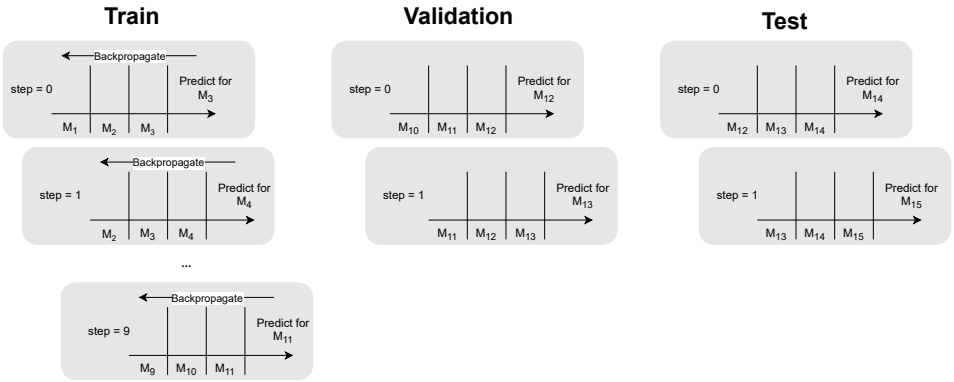


Figure A.1: Train-validation-test split and model training. Windows are obtained by 1-month shift. Backpropagation happens at the end of the time window, and the prediction is made for the last month of the window.

Table A.1: Edge types

Type	Description	Data source
Credit card	There exist an edge between users who used the same credit card in the app.	Delivery app
References	There exist an edge between users who referenced/got referenced	Credit card usage
Geohash	There exist an edge between users who ordered more than 4 times in the close geographical proximity (Geohash 7 - 152.9m x 152.4m)	Delivery app
Contacts	There exist an edge between users if one of them have the other one in its phone contacts	Delivery app

Table A.2: Best hyperparameter settings found by grid search. Static hyperparameters: epochs = 100, learning rate = 0.001, optimizer = ADAM [Kingma and Ba, 2014]

Model	#hiden dimensions RNN	#embeddings GNN	#layers GNN	SMOTE sample rate
GCN+LSTM	400	300	6	0
GCN+GRU	200	300	2	0
GAT+LSTM	300	400	2	0
GAT+GRU	200	200	2	0
PageRank+LSTM	200	\	\	0.75
PageRank+GRU	200	\	\	0
Features+LSTM	300	\	\	0
Features+GRU	300	\	\	0
Static GAT	\	300	4	0
Static GCN	\	300	4	0.75

Table A.3: Resource types

Resource	Specification
Memory per CPU	250G
CPU cores per task	2
Processor	Intel E5-2683 v4 Broadwell @ 2.1GHz