Prediction of the effects of epidemic spreading with graph neural networks

Sebastian Mežnar[1], Nada Lavrač[1,2], and Blaž Škrlj[1,2]

[1] Jožef Stefan Institute, Jamova 39, Ljubljana, Slovenia
[2] Jožef Stefan International Postgraduate School, Jamova 39, Ljubljana, Slovenia

Abstract. Understanding how information propagates in real-life complex networks yields a better understanding of dynamical processes such as misinformation or epidemic spreading. With the recent resurgence of graph neural networks as a powerful predictive methodology, many network properties can be studied in terms of their predictability and as such offer a novel view on the studied process, with the direct application of fast predictions that are complementary to resource-intensive simulations. We investigated whether graph neural networks can be used to predict the effect of an epidemic, should it start from a given individual (patient zero). We reformulate this problem as node regression and demonstrate the high utility of network-based machine learning for a better understanding of the spreading effects. By being able to predict the effect of a given individual being the patient zero, the proposed approach offers potentially orders of magnitude faster risk assessment and potentially aids the adopted epidemic spreading analysis techniques.

Keywords: epidemics, neural networks, machine learning, spreading

1 Introduction

The spread of information and disease is a common phenomenon that has a lot of practical and sometimes life-saving applications. One of these applications is the creation of better strategies for stopping the spreading of misinformation on social media or an epidemic. Further, companies can analyze spreading to create better strategies for marketing their product [6, 19]. Spreading analysis can also be suitable for analysis of e.g., fire spreading, implying large practical value in terms of insurance cost analysis [8]. Analysis of spreading is commonly studied via extensive simulations [11]. Here, the ideas from statistical mechanics are exploited to better understand both the extent of spreading, as well as its speed [2].

Albeit offering high utility, reliable simulations of spreading processes can be *expensive* on larger networks, which we believe can be addressed by the employment of *machine learning*-aided modelling [29]. The contributions of this work are multifold and can be summarized as follows.

1. We re-formulate the task of identification of the spreading effect from a given node into a node regression problem.

2. The prediction problem is addressed with state-of-the-art graph neural network-based approaches, as well as a simpler, centrality-based approach proposed as a part of this work.
3. Consistent predictive capability is demonstrated across multiple real-life networks, demonstrating that graph neural network-based approaches can serve as a complementary, highly efficient analysis tool when studying information spreading.
4. A methodology is proposed that performs notably better than the random baseline on the datasets we tested.
5. We demonstrate how individual predictions of the obtained models can be *explained* via the game-theoretic explanation tool SHAP [17].

The remainder of this work is structured as follows. In Section 2, we discuss the related work which led to the proposed approach. Next, we re-formulate the task and show its importance in Section 3. We propose a new approach based on node centralities to solve the re-formulated task in Section 4. In Section 5 we present the datasets, experimental setting and results of our empirical evaluation. We conclude the paper in Section 6.

2 Related work

In the following section, we discuss the relevant related work. We begin by discussing the notion of contagion processes, followed by an overview of graph neural networks.

2.1 Analysis of spreading processes

The study of spreading processes on networks is a lively research endeavour [19]. Broadly, spreading processes can be split into two main branches, namely, the simulation of *epidemics* and *opinion dynamics*. The *epidemic spreading* models can be classical or network-based. Here, the classical models are for example systems of differential equations that do not account for a given network's topology. Throughout this work, we are interested in extensions of such models to real-life network settings. One of the most popular spreading models on networks is the Susceptible-Infected-Recovered (SIR) [10] model. The spread of the pandemic in the SIR model is dictated by parameters β known as the infection rate and γ known as the recovery rate. Nodes in this model can have one of three states (Susceptible, Infected, Recovered).

SIR assumes that if a susceptible node comes into contact with an infected one during a generic iteration, it becomes infected with probability β. In each iteration after getting infected, a node can recover with probability γ (the only transition allowed are S to I to R).

Other related models include, for example, SEIR, SEIS, SWIR[3]. Further, one can also study the role of cascades [9] or the Threshold model [4].

[3] Where S-Susceptible, I-Infected, R-Recovered, E-Exposed and W-Weakened.

2.2 Machine learning on networks

Learning by propagating information throughout a given network has already been considered by the approaches such as label propagation [30]. However, in recent years, the approaches that jointly exploit both the adjacency structure of a given network alongside features assigned to its nodes are becoming prevalent in the network learning community. The so-called graph neural networks have re-surfaced with the introduction of the Graph Convolutional Networks (GCN) [13]; an idea where the normalized adjacency matrix is directly multiplied with the feature space and effectively represents a neural network layer. Multiple such layers could be stacked to obtain better approximation power/performance. One of the most recent methods from this branch are the Graph Attention Networks [28], an extension of GCNs with the notion of neural *attention*. Here, part of the neural network focuses on particular parts of the adjacency space, offering more robust and potentially better performance.

Albeit being in widespread use, graph neural networks are not necessarily the most suitable choice when learning solely from the network adjacency structure. For such tasks, methods such as node2vec [5], SGE [26] and DeepWalk [21] were developed. This branch of methods corresponds to what we refer to as *structural* representation learning. In our work, we focus mostly on learning this type of representations using network centrality information.

Note that although graph neural networks are becoming the prevailing methodology for learning from *feature-rich* complex networks, it is not clear whether they perform competitively to the more established structural methods if the feature space is derived solely from a given network's structure.

3 Task formulation

When analysing an epidemic there are three main pieces of information that give us most insight about how severe an epidemic was. The first crucial information is when an epidemic reaches the peak (most nodes infected) since we are less likely to be able to stop an epidemic when the peak is reached too quickly. This information is especially important when trying to find a cure for a disease or trying to stop misinformation on social media. Related to this, we usually want to know, how many nodes will be infected when the epidemic reaches its peak. When the maximum number of people infected by some disease is high, there might not be enough beds or medicine for everyone. In contrast, companies might want to create marketing campaigns on platforms such as Twitter and target specific users to reach a certain number of retweets that are needed to become trending. Another important insight into an epidemic is how many people get infected. If a scam on the internet reaches a lot of people there is a greater chance that more people will fall for it.

In our work, we focus on predicting the maximum number of infected nodes and the time this number is reached. We create target data by simulating epidemics from each node with SIR diffusion model and identifying the number, as

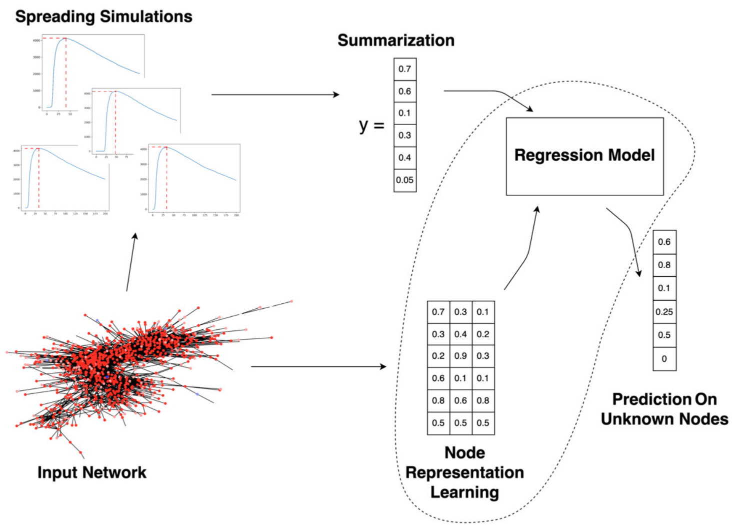

Fig. 1: Overview of the proposed methodology.

well as the time when the maximum number of nodes are infected. We aggregate the generated target data by taking the mean values for each node. In the end, we preprocess this data by normalizing it.

4 Proposed methodology

In this section, we present the creation of target data and summarize centralities and learners used for the regression task. An overview of the proposed methodology can be seen in Figure 1. The figure shows two branches. On the upper branch, simulations are created and transformed into target data, while the node representation is learned on the lower branch. After this, a regression model is trained with data from both branches and used to generate predictions for the remaining (unknown) nodes.

The initial part of the methodology addresses the issue of input data generation. Intuitively, the first step simulates epidemic spreading from individual nodes of a given network to assess both the time required to reach the maximum number of infected, as well as the number itself. In this work, we leverage the widely used SIR model [10] to simulate epidemics, formulated as follows.

$$\frac{dS}{dt} = -\frac{\beta \cdot S \cdot I}{N}$$
$$\frac{dI}{dt} = \frac{\beta \cdot S \cdot I}{N} - \gamma \cdot I$$
$$\frac{R}{dt} = \gamma \cdot I,$$

where S represents the number of susceptible, R the number of recovered and I the number of infected individuals. Spreading is governed by input parameters γ and β. The SIR model is selected due to many existing and optimized implementations that are adapted from systems of differential equations to networks [6]. We use NDlib [23] to simulate epidemics based on the *SIR diffusion model*.

Target data creation results in two real values for each node. We attempt to *predict* this two values. The rationale for the construction of such predictive models is, they are potentially much faster than simulating multiple processes for each node (prediction time is the bottleneck) and can give more insight into *why* some nodes are more "dangerous". The predictive task can be formulated as follows. Let $G = (V, E)$ represent the considered network. We are interested in finding the mapping $f : V \rightarrow \mathbb{R}^+$, such that this mapping maps from the set of nodes V to the set of real values that represent e.g., the maximum number of infected individuals if the spreading process is started from a given node $u \in V$. Thus, f corresponds to **node regression**.

The models we considered can broadly be split into two main categories; graph neural networks and propositional learners. The main difference between the two is that the graph neural network learners, such as GAT [28] and GIN [31] simultaneously exploit both structure of a network, as well as *node features*, whilst the propositional learners take as input only the constructed feature space (and not the adjacency matrix). As an example, the feature space is passed throughout the GIN's structure via the update rule that can be stated as:

$$\boldsymbol{h}_v^{(k)} = \mathrm{MLP}^{(k)}\left(\left(1 + \epsilon^{(k)}\right) \cdot \boldsymbol{h}_v^{(k-1)} + \sum_{\boldsymbol{u} \in V(v)} \boldsymbol{h}_u^{(k-1)}\right),$$

where MLP corresponds to a multilayer perceptron, ϵ a hyperparameter, $\boldsymbol{h}_u^{(k)}$ the node u's representations at layer k and $V(v)$ the v-th node's neighbors. We test both graph neural networks and propositional learners as it is to our knowledge not clear, whether direct incorporation of the adjacency matrix offers any superior performance, as the graph neural network models are computationally more expensive. The summary of considered learners is offered in Table 1.

As the considered complex networks do not possess *node attributes*, we next discuss which features, derived solely from network structure were used in the considered, state-of-the-art implementations of GAT [28] and GIN [31]. Further, we also test models where only the constructed structural features are considered, as well a standalone method capable of learning node representations, combined with the XGBoost [1] classifier. In this work, we explored whether *centrality-based* descriptions of nodes are suitable for the considered learning task. The rationale for selecting such features is, they are potentially fast to compute and entail global relation of a given node w.r.t. the remaining part of the networks. The centralities, computed for each node are summarized in Table 2. After calculating these centralities, we normalize and concatenate them to create the final features. This features together with XGBoost classifier are referred to as CABoost in Section 5.3, which is considered one of the contributions of this work.

Table 1: Summary of the considered learners with descriptions. Here, $\boldsymbol{A}$ denotes the adjacency matrix and $\boldsymbol{F}$ the feature matrix.

Input	Learner	Method description
$\boldsymbol{A}, \boldsymbol{F}$	GAT	Graph Attention Networks
$\boldsymbol{A}, \boldsymbol{F}$	GIN	Graph Isomorphism Networks
$\boldsymbol{A}$	node2vec + XGBoost	node2vec-based features + XGBoost
$\boldsymbol{F}$	CABoost (our)	XGBoost trained solely on centrality based features

Table 2: Summary of the centralities considered in our work.

Centrality	Time complexity	Description
Degree centrality [22]	$\mathcal{O}(\|E\|)$	The number of edges of a given node
Eigenvector centrality [22]	$\mathcal{O}(\|V\|^3)$	Importance of the node, where nodes are more important if they are connected to other important nodes. This can be calculated using the eigenvectors of the adjacency matrix.
PageRank [20]	$\mathcal{O}(\|E\|)$	Probability that a random walker is at a given node.
Average Out-degree	$\mathcal{O}(\|V\| \cdot w \cdot \overline{s})$	The average out-degree of nodes encountered during w random walks of mean length $\overline{s}$
Hubs and Authorities (HITS) [14]	$\mathcal{O}(\|E\|)$	HITS is a link analysis algorithm that assigns two scores to each node. Authority score represents how important a node is and the hub score represents how well a node is connected to other important nodes.

During model training we minimized the mean squared error between the observed states ($f(u)$) and the predictions (y_u); stated for the u-th node as

$$MSE = 1/|N| \sum_{u \in N} (f(u) - y_u)^2.$$

To summarize, we learn network features with fast algorithms and use them together with GIN, GAT and XGBoost learners to minimize the mean squared error between predictions and data we make using simulations on part of the network. We next present the evaluation process and results of this methodology.

5 Empirical evaluation

In this section, we show the empirical results of our approach and compare it to other baselines. We also present how predictions from CABoost model can be explained using SHAP [16].

5.1 Baselines

We compared the results of proposed method to the following 4 baselines:

- *Random baseline* creates an embedding of size $|N| \times 64$ with random numbers drawn from Unif$(0, 1)$.
- *node2vec* [5] learns a low dimensional representation of nodes that maximizes the likelihood of neighborhood preservation using random walks. During testing, we use the default parameters.
- *GAT* [28] includes attention mechanism that helps learn the importance of neighboring nodes. In our tests, we use the implementation from PyTorch Geometric [3].
- *GIN* [31] learns a representation that can provably achieve the maximum discriminative power. In our tests, we use the implementation from PyTorch Geometric [3].

For comparison we also add the averaged simulation error. We calculate this error with the MSE formula, where we use the mean absolute difference between the value we get from simulations and the mean value for that node as $f(u)$ and 0 as y_u. This baseline corresponds to a situation, where only a single simulation would be used to approximate the expected value of multiple ones (the goal of this work).

5.2 Experimental setting

For testing[4], we used datasets: Hamsterster [7], Advogato [18], Wikipedia Vote [15] and FB Public Figures [25], taken from the Network Repository website [24]. Some basic statistics of the networks we used can be seen in Table 3. Two networks used during testing are visualized in Figure 2. The network nodes in this figure are colored based on the values of the target variables.

Table 3: Basic statistics of the networks used for testing.

Name	Nodes	Edges	Components
Wikipedia Vote [15]	889	2914	1
Hamsterster [7]	2426	16630	148
Advogato [18]	6551	43427	1441
FB Public Figures [25]	11565	67114	1

We used the following approach to test the proposed method as well as baselines mentioned in Section 5.1. We created the target data by simulating five epidemics starting from each node of every dataset. We created each simulation using the SIR diffusion model from the NDlib [23] Python library with parameters $\beta = 5\%$ and $\gamma = 0.5\%$. We then create the target variables by identifying

[4] That can be found at https://github.com/smeznar/Epidemic-spreading-CN2020.

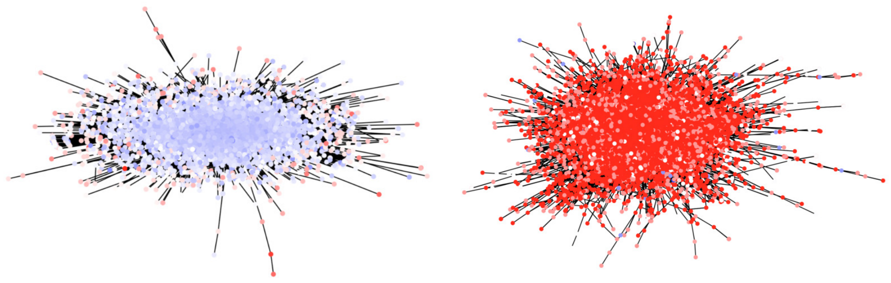

Fig. 2: Visualization of Advogato (left) and FB Public Figures (right) networks. The color represents the target value we get when starting the spreading from a given node. Color on Advogato dataset represents the time needed to reach the peak while on FB Public Figures dataset maximum number of infected nodes is shown. Blue colors represent low values while red ones represent high ones.

and aggregating the maximum number of infected nodes and the time when this happens. We use these variables to test methods using five-fold cross-validation. We used XGBoost [1] with default parameters as the regression model with proposed features based on the mentions centralities, the random baseline and the node2vec [5] baseline. Baselines GIN and GAT were trained for 200 epochs using the Adam optimizer [12].

5.3 Results

The results of the evaluation described in Section 5.2 are presented in Tables 4 and 5. We can see that in most cases both time and the maximum number of infected nodes can be predicted significantly better by using information about the structure of the network.

Observing results in Table 4 we see that overall CABoost achieves best results and is beaten only on the Wiki vote dataset by GAT and the Hamsterster dataset by GIN. We can further see that although node2vec does not achieve the best score on any dataset, it consistently achieves results that are comparable to CABoost. The biggest improvement over the random baseline can be seen on datasets Hamsterster and Advogato that have more than one connected components. We can also see that only GAT achieves a result that is significantly better than the random baseline on the Wiki vote dataset.

The results in Table 5 are very similar to those of showcased in Table 4. Here CABoost performs even better and is outperformed only by GIN on the Wiki vote dataset. We can see that when predicting time, random baseline performs significantly worse than all other baselines on all datasets but that overall these predictions are better than when the maximum number of infected nodes is being predicted.

We can see on all datasets that prediction with such learners is more beneficial than creating only one simulation, further showing their usefulness.

Table 4: Cross-validation results for maximum number of infected node.

	Wiki vote	Hamsterster	Advogato	FB public figures
Random	0.0191 (±0.0046)	0.1633 (±0.0123)	0.2052 (±0.0055)	0.0144 (±0.0014)
node2vec+XGBoost	0.0200 (±0.0034)	0.0060 (±0.0019)	0.0073 (±0.0010)	0.0127 (±0.0009)
GAT	**0.0149 (±0.0039)**	0.0460 (±0.0015)	0.0653 (±0.0079)	0.0129 (±0.0009)
GIN	0.0234 (±0.0078)	**0.0042 (±0.0006)**	0.0253 (±0.0195)	0.0116 (±0.0007)
CABoost (our)	0.0187 (± 0.0040)	0.0045 (±0.0012)	**0.0067 (±0.0012)**	**0.0114 (±0.0011)**
Simulation error (averaged)	0.0486 (± 0.0481)	0.0083 (± 0.0223)	0.0107 (±0.0251)	0.0546 (±0.0375)

Table 5: Cross-validation results for time when most nodes are infected.

	Wiki vote	Hamsterster	Advogato	FB public figures
Random	0.0168 (±0.0015)	0.0168 (±0.0013)	0.0390 (±0.0014)	0.0094 (±0.0004)
node2vec+XGBoost	0.0126 (±0.0010)	0.0062 (±0.0005)	0.0053 (±0.0007)	0.0054 (±0.0004)
GAT	0.0118 (±0.0010)	0.0125 (±0.0008)	0.0190 (±0.0015)	0.0066 (±0.0003)
GIN	**0.0096 (±0.0012)**	0.0045 (±0.0005)	0.0126 (±0.0100)	0.0045 (±0.0005)
CABoost (our)	0.0103 (±0.0017)	**0.0044 (±0.0007)**	**0.0038 (±0.0007)**	**0.0042 (±0.0003)**
Simulation error (averaged)	0.0168 (± 0.0630)	0.0063 (± 0.0467)	0.0066 (±0.0352)	0.0093 (±0.0424)

5.4 Interpretation of a prediction

We can explain predictions using model explanation approaches such as SHapley Additive exPlanations (SHAP) [16, 27]. SHAP is a game-theoretic approach for explaining classification and regression models. The algorithm perturbs subsets of input features to take into account the interactions and redundancies between them. The explanation model can then be visualized, showing how the feature values of an instance impact a prediction.

An example of such an explanation is shown in Figure 3. We can see that both HITS centralities do not impact the explanation much. We can also see that small values of PageRank impact the prediction positively, while small values of Eigenvector and Degree centrality impact the prediction negatively. The above-average out-degree centrality also impacts the model negatively.

6 Discussion and conclusions

In this paper, we showcase how contemporary graph neural network-based methods can be used for fast estimation of epidemic spreading effect from a given node. We showed that by re-formulating the task as node regression, we can obtain realistic estimates much faster than by performing computationally expensive simulations, even though such simulations are initially used to fine-tune the machine learning models. Further, employment of predictive modeling instead on relying on a single simulation also showed promising results.

We show that while graph neural networks outperform the random baseline and sometimes give us great results, centrality scores and node2vec feature representation coupled with XGBoost mostly outperform them. We also see that machine learning models might overall give a more accurate representation of an epidemic than data gathered from a small number of simulations.

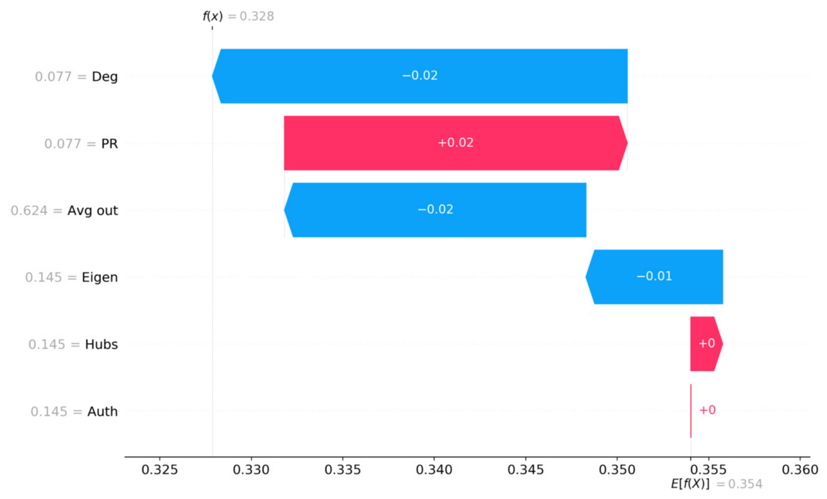

Fig. 3: An example of a model explanation for an instance. Blue arrows indicate how much the prediction is lowered by some feature value, while the red ones indicate how much it is raised. Prediction starts at models expected value 0.354 and finishes at 0.328. Features and their values are shown on the left. The visualization shows for example that the prediction dropped from 0.350 to 0.328 because of the low value of degree centrality.

An obvious limitation of the proposed task is that the spreading is probabilistic and even the best classifiers might make significant errors. Similarly when observing prediction results of the maximum number of infected nodes one must be careful since we predict an average outcome from some node and not the true maximum. This gives us the ability to predict which nodes are most "dangerous" as patient zero. When trying to predict an outcome of an epidemic that has already spread, one must adjust data accordingly and get rid of simulations where epidemics have not spread.

In further work, we plan to research different centralities and algorithms to better describe network structure and achieve more accurate results. Another aspect of our interest is how the proposed method scales and how well it works on different types of networks. We also plan to further research the ability to solve such tasks by using unsupervised algorithms.

7 Acknowledgments

The work of the last author (BŠ) was funded by the national research agency (ARRS)'s grant for junior researchers. The work of other authors was supported by the Slovenian Research Agency (ARRS) core research program P2-0103 and P6-0411, and research projects J7-7303, L7-8269, and N2-0078 (financed under the ERC Complementary Scheme).

References

1. Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.
2. Suyalatu Dong, Feng-Hua Fan, and Yong-Chang Huang. Studies on the population dynamics of a rumor-spreading model in online social networks. *Physica A: Statistical Mechanics and its Applications*, 492:10–20, 2018.
3. Matthias Fey and Jan E. Lenssen. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.
4. Mark Granovetter. Threshold Models of Collective Behavior. *American Journal of Sociology*, 83(6):1420–1443, 1978.
5. Aditya Grover and Jure 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.
6. Adrien Guille, Hakim Hacid, Cecile Favre, and Djamel A. Zighed. Information Diffusion in Online Social Networks: A Survey. *SIGMOD Rec.*, 42(2):17–28, July 2013.
7. Hamsterster. Hamsterster social network. http://www.hamsterster.com.
8. Ahmed Kacem, Christine Lallemand, Nathalie Giraud, Maxime Mense, Matthieu De Gennaro, Yannick Pizzo, Jean-Claude Loraud, Pascal Boulet, and Bernard Porterie. A small-world network model for the simulation of fire spread onboard naval vessels. *Fire Safety Journal*, 91:441–450, 2017.
9. David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the Spread of Influence through a Social Network. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, page 137–146, New York, NY, USA, 2003. Association for Computing Machinery.
10. William Ogilvy Kermack, A. G. McKendrick, and Gilbert Thomas Walker. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 115(772):700–721, 1927.
11. Sergey Kesarev, Oksana Severiukhina, and Klavdiya Bochenina. Parallel simulation of community-wide information spreading in online social networks. In *Russian Supercomputing Days*, pages 136–148. Springer, 2018.
12. Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2015.
13. Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations (ICLR)*, 2017.
14. Jon M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. *J. ACM*, 46(5):604–632, September 1999.
15. Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Signed networks in social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1361–1370. ACM, 2010.
16. Scott M Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

17. Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774, 2017.
18. Paolo Massa, Martino Salvetti, and Danilo Tomasoni. Bowling alone and trust decline in social network sites. In *Dependable, Autonomic and Secure Computing, 2009. DASC'09. Eighth IEEE International Conference on*, pages 658–663. IEEE, 2009.
19. Cameron Nowzari, Victor M Preciado, and George J Pappas. Analysis and control of epidemics: A survey of spreading processes on complex networks. *IEEE Control Systems Magazine*, 36(1):26–46, 2016.
20. L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. In *WWW 1999*, 1999.
21. Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. ACM.
22. Francisco Aparecido Rodrigues. Network Centrality: An Introduction. *A Mathematical Modeling Approach from Nonlinear Dynamics to Complex Systems*, page 177, 2019.
23. Giulio Rossetti, Letizia Milli, Salvatore Rinzivillo, Alina Sîrbu, Dino Pedreschi, and Fosca Giannotti. NDlib: a python library to model and analyze diffusion processes over complex networks. *International Journal of Data Science and Analytics*, 5(1):61–79, 2018.
24. Ryan A. Rossi and Nesreen K. Ahmed. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*, 2015.
25. Benedek Rozemberczki, Ryan Davies, Rik Sarkar, and Charles Sutton. GEMSEC: Graph Embedding with Self Clustering. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2019*, pages 65–72. ACM, 2019.
26. Blaž Škrlj, Nada Lavrač, and Jan Kralj. Symbolic Graph Embedding Using Frequent Pattern Mining. In Petra Kralj Novak, Tomislav Šmuc, and Sašo Džeroski, editors, *Discovery Science*, pages 261–275, Cham, 2019. Springer International Publishing.
27. Erik Štrumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3):647–665, 2014.
28. Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018.
29. Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.
30. Zhu Xiaojin and Ghahramani Zoubin. Learning from labeled and unlabeled data with label propagation. *Tech. Rep., Technical Report CMU-CALD-02–107, Carnegie Mellon University*, 2002.
31. Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*, 2019.