2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA) | 978-1-6654-7330-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/DSAA54385.2022.10032349

Improving Source Localization by Perturbing Graph Diffusion

Yaping Zhao[1,2]
zhaoyp@connect.hku.hk

Zhongrui Wang[1,2]
zrwang@eee.hku.hk

Edmund Y. Lam[1,2,*]
elam@eee.hku.hk

[1]*The University of Hong Kong*

[2]*ACCESS — AI Chip Center for Emerging Smart Systems*

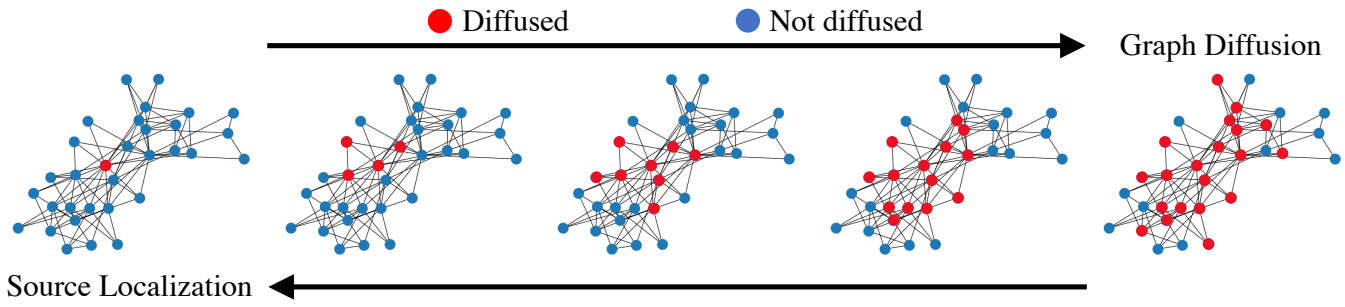


Fig. 1: From left to right, the graph diffusion phenomena that information propagates along the vertex connections to adjacent nodes. From right to left, the graph source localization that identifies those nodes from which the information started to spread.

***Abstract*—Graph diffusion has quite common phenomenons in our daily life, such as misinformation propagation. As the inverse problem of graph diffusion, the goal of source localization is to identify those nodes of the network from which the information started to spread. Though graph diffusion has been well explored in the literature, the emerging source localization problem is important yet challenging because of its intrinsic ill-posed characteristics. While graph neural networks (GNN) are recently utilized to implement source localization and achieve state-of-the-art performance, a general GNN framework consists of two stages: feature construction and label propagation. Typically, a neural network is pretrained in the feature construction, and then combine with additional functions to jointly perform finetuning for source localization. However, those emerging methods have risks in overfitting the feature construction task, which usually has a gap with the target downstream task of source localization. Such a gap is neglected by previous methods and leads to suboptimal performance. To address this issue, we propose a very simple yet effective method to help better finetune feature construction on the source localization task by adding some noise to the parameters of the feature construction model before finetuning. More specifically, we utilize a matrix-wise perturbing method that adds different uniform noises to different parameter matrices, and design the noise considering the variances and magnitude of network weights. Extensive experiments on six real-world datasets show the proposed method can consistently empower the finetuning of different pretrained feature construction models on the downstream source localization task. Moreover, we conduct an ablation study to investigate the performance with different noise types and intensities. Code is available at: https://github.com/IndigoPurple/PGD.**

*Corresponding author.

This work is supported in part by ACCESS — AI Chip Center for Emerging Smart Systems, Hong Kong SAR, in part by Hong Kong Research Grant Council (Grant No. 27206321), National Natural Science Foundation of China (Grant No. 62122004).

***Index Terms*—Graph Diffusion, Source Localization, Inverse Problem, Social Networks, Information Systems**

I. Introduction

Since graph networks have developed a wide spectrum of applications in various domains [1]–[3], information diffusion within the graph networks has attracted broad research interests [4], [5]. Recently, the inverse problem of information diffusion leads to an emerging and interesting task, *i.e.*, source localization. As Figure 1 shows, graph diffusion is the phenomenon in which information propagates along with the vertex connections to adjacent nodes, while the source localization identifies those nodes from which the information started to spread. Graph source localization also covers a wide range of promising research and real-world applications. For instance, tracking rumors to the source in real time is a possible way to dispel misinformation and reduce stigma [6]; identifying the source Emails carrying viruses in the online networks offer prevention of computer virus epidemics [7]; source localization enables us to build a detection system against malware by locating the malware source in the Internet of Things (IoT) network [8].

With the rapid advancement of Graph Neural Networks (GNN) [9], [10], graph diffusion has been well explored. However, its inverse problem, *i.e.*, the source localization, is much more challenging. Such an inverse problem is intrinsically ill-posed, because of the fundamental issue as Figure 2 shows: different source nodes could generate the same cascade pattern. Although the learned knowledge from graph diffusion models could facilitate predicting which node is likely to be the source, it is prohibitively hard to incorporate graph

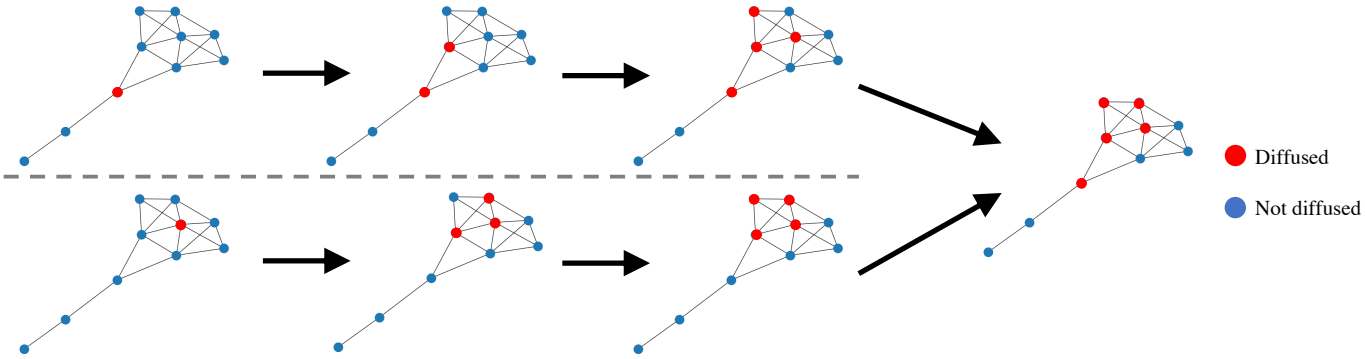


Fig. 2: As the inverse problem of graph diffusion, source localization is difficult because different source nodes could generate the same cascade pattern.

diffusion models into the inverse problem directly and achieve source localization since they are opposite processes.

Traditional methods [11]–[14] usually assume the topologies of networks and apply the classical probabilistic graphical models, thus are limited to a specific type of neural network and are difficult to generalize. With the development of deep learning and neural networks, graph convolutional networks [15] and invertible graph diffusion neural networks [16] are proposed for generic end-to-end source localization. While there are many existing GNN-based graph diffusion models, a general GNN framework consists of two stages: feature construction and label propagation. In the feature construction, a neural network is pretrained to estimate the initial node diffusion vector. In the label propagation, a propagation function is designed to diffuse information to neighboring nodes. However, those emerging methods have risks in overfitting the feature construction task, which usually has a gap with the target downstream task of source localization. Such a gap is neglected by previous methods and leads to suboptimal performance.

In this paper, inspired by a recent work NoisyTune [17] for natural language processing (NLP), we propose a very simple yet effective method named PGD to help better finetune feature construction on the source localization task by adding some noise to the parameters of the feature construction model before finetuning. More specifically, we utilize a matrix-wise perturbing method which adds different uniform noises to different parameter matrices. To take a further step from NoisyTune, we design the noise considering the variances and magnitude of network weights, and conduct an ablation study to investigate the performance with different noises. Extensive experiments on six real-world datasets show the proposed method can consistently empower the finetuning of different pretrained feature construction models on the downstream source localization task.

In a nutshell, our main contributions are as follows:

- We propose a method named PGD to improve the performance of source localization by perturbing the feature construction for graph diffusion before finetuning on the source localization task.
- Extensive experiments on six real-world datasets demonstrate the effectiveness of our proposed method and show superior performance against state-of-the-art algorithms.
- The ablation study provides a further investigation of the performance improvement regarding different noise types and intensities.

II. Related Work

A. Graph Diffusion

Graph diffusion aims at predicting the information dissemination in complex networks, which has been widely applied in real-world applications such as societal event prediction [18], and adverse event detection in social media [19], [20]. Traditional methods [11]–[14] usually assume the topologies of networks and apply the classical probabilistic graphical models, thus are limited to a specific type of neural network and are difficult to generalize. A recent line of research works usually include multimodality [21], and various techniques have been applied including self-attention mechanism [22], knowledge base [23], multi-task learning [24], and stochastic processes [25]–[27]. However, they cannot utilize network topology to enhance predictions. To address this issue, GNN has been combined with RNN [28]–[30], and a handful of other neural network architectures are also investigated [31], [32].

B. Graph Source Localization

The goal of the graph source localization is to identify the source of a network based on observations such as the states of the nodes and a subset of timestamps at which the diffusion process reached the corresponding nodes [16], [33]. Graph source localization has a wide range of applications such as disease localization, virus localization, and rumor detection [34]–[36]. Similar to graph diffusion models, existing graph source localization papers usually require the assumptions of the diffusion, network topology, and observations. With the development of GNNs, Dong et al. proposed a Graph Convolutional Network for multiple source localization [15]. However, its model relies heavily on hand-crafted rules. To break through the limits of previous methods, Zhao *et*

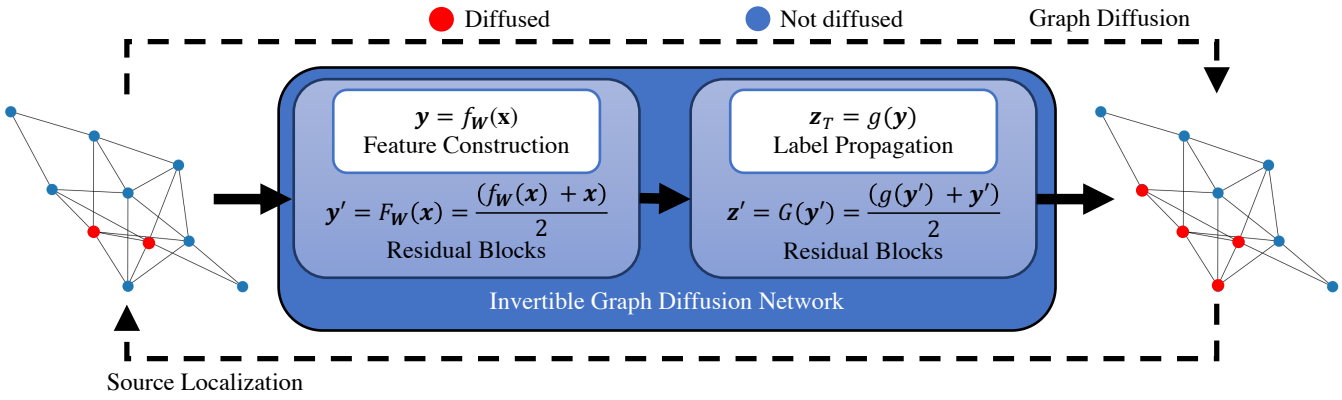


Fig. 3: Overview of the invertible graph diffusion network. While a GNN-based graph diffusion framework generally consists of two stages, feature construction and label propagation, we adopt the network from IVGD, which consists of graph residual blocks to model invertible graph diffusion.

Notations	Descriptions
V	The node set.
E	The edge set.
$\boldsymbol{x}$	The source node vector.
$\boldsymbol{y}$	The initial node diffusion vector.
$\boldsymbol{z}_t$	The diffusion vector at time t.
T	The length of diffusion.
$f_{\mathbf{W}}$	The function of feature construction.
g	The function of label propagation.
$\Phi(\boldsymbol{x}) = 0$	The equality constraint of a validity pattern.

TABLE I: Key notations and descriptions.

al. [16] proposed an invertible graph diffusion neural network for source localization, named IVGD, which consists of graph residual blocks of feature construction and label propagation to model invertible graph diffusion.

It is worth noting that all the aforementioned GNN framework generally consists of two stages: feature construction and label propagation. In the feature construction, a neural network is learned to estimate the initial node diffusion vector. In the label propagation, a propagation function is designed to diffuse information to neighboring nodes. However, those emerging methods have risks in overfitting the feature construction task, which usually has a gap with the target downstream task of source localization. Such a gap is neglected by previous methods and leads to suboptimal performance.

III. Method

In this section, we will first formulate the source localization problem in Section III-A and describe the invertible graph diffusion network in Section III-B, as Figure 3 shows. Next, we will devise a very simple yet effective method named in Section III-C to help better finetune feature construction on the source localization task, as Figure 4 shows.

A. Problem Statement

As source localization aims to inversely infer sources of graph diffusion, it could be formulated mathematically in the form of an inverse problem. Table I outlines key notations used in this paper. Consider a graph $G = \{V, E\}$, where $V = \{v_1, \ldots, v_n\}$ and E are the node set and the edge set respectively, $|V| = n$ is the number of nodes. $\boldsymbol{z}_t \in \{0, 1\}^n$ is a diffusion vector at time t. $\boldsymbol{z}_{t,i} = 1$ means that node i is diffused at time t, while $\boldsymbol{z}_{t,i} = 0$ means that node i is not diffused at time t. S is a set of source nodes. $\boldsymbol{x} \in \{0, 1\}^n$ is a vector of source nodes, $\boldsymbol{x}_i = 1$ if $v_i \in S$ and $\boldsymbol{x}_i = 0$ otherwise. The diffusion process begins at timestamp 0 and terminates at timestamp T. While there are many existing GNN-based graph diffusion models, a general GNN framework consists of two stages: feature construction and label propagation.

In the feature construction, a neural network $f_{\mathbf{W}}$ is learned to estimate the initial node diffusion vector based on input $\boldsymbol{x}$:

$$\boldsymbol{y} = f_{\mathbf{W}}(\boldsymbol{x}), \tag{1}$$

where $\mathbf{W}$ is the learnable weights in $f_{\mathbf{W}}$.

In the label propagation, a propagation function g is designed to diffuse information to neighboring nodes:

$$\boldsymbol{z}_T = g(\boldsymbol{y}). \tag{2}$$

Therefore, the graph diffusion model is:

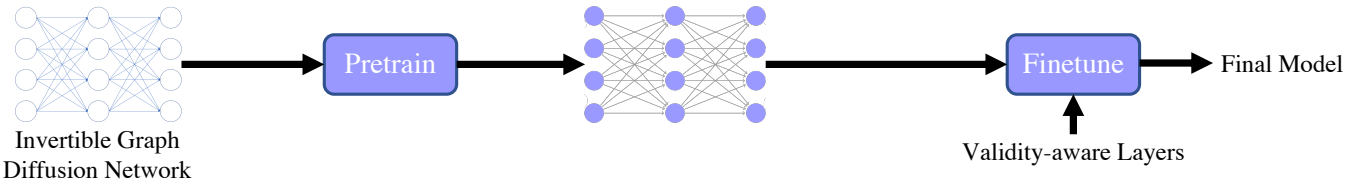
$$p = g(f_{\mathbf{W}}(\boldsymbol{x})), \tag{3}$$

and its inverse problem, graph source localization, is to infer $\boldsymbol{x}$ from $\boldsymbol{z}_T$. Moreover, a validity pattern can be imposed on sources in the form of the constraint $\Omega(\boldsymbol{x}) = 0$ such as the number of source nodes, and the connectivity among multiple sources. Then the graph source localization problem can be mathematically formulated as follows:

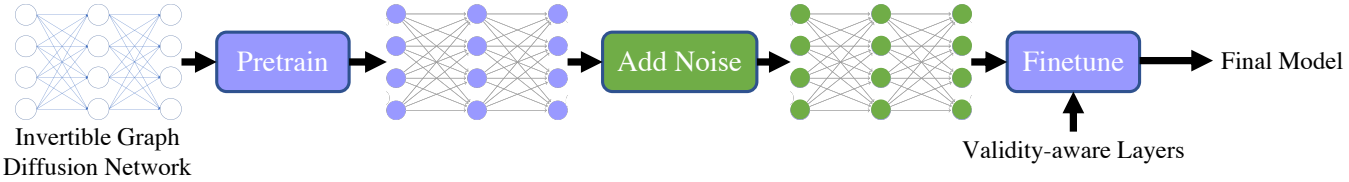
$$p^{-1} : \boldsymbol{z}_T \to \boldsymbol{x} \qquad s.t. \Omega(\boldsymbol{x}) = 0. \tag{4}$$

B. Invertible Graph Diffusion Network

Our goal in this subsection is to obtain an approximate estimation of the source vector $\boldsymbol{x}$ based on the learned knowledge from the graph diffusion model p. One intuitive idea is to invert the process of the forward model p. The key challenge here is



(a) Standard IVGD finetuning.



(b) Our method.

Fig. 4: Schematic comparisons between standard IVGD finetuning and our method.

that p is not necessarily invertible, so the task is how to devise an invertible architecture based on p.

Now we focus on determining an invertible GNN-based architecture. While there are many classic invertible architectures such as iRevnet and Glow [37]–[39], their forms are quite complex and require extra components to ensure one-to-one mapping. IVGD [16], which extended i-ResNet [40] to the GNN, stands out among others because of its simplicity and outstanding performance. We adopt IVGD as our invertible architecture, and Figure 3 provides the overview of our invertible graph diffusion network. Specificallt, we first formulate the graph residual net of the general GNN framework:

$$\boldsymbol{y}' = F_{\mathbf{W}}(\boldsymbol{x}) = (f_{\mathbf{W}}(\boldsymbol{x}) + \boldsymbol{x})/2, \tag{5}$$

$$\boldsymbol{z}'_T = G(\boldsymbol{y}') = (g(\boldsymbol{y}') + \boldsymbol{y}')/2, \tag{6}$$

where $F_{\mathbf{W}}(\cdot)$ and $G(\cdot)$ are graph residual blocks of feature construction and label propagation, respectively. The graph residual net for graph diffusion could be denoted as :

$$P(\boldsymbol{x}) = G(F_{\mathbf{W}}(\boldsymbol{x})), \tag{7}$$

and its inverse for graph source localization could be denoted as P^{-1}. Here, P can be inversed to P^{-1} by fixed point iterations [16].

C. Pertubring Graph Diffusion

As Figure 4(a) shows, a standard IVGD will typically first pretrain the neural network on the feature construction task, and then incorporate it with the validity-aware layers to conduct finetuning for the source location task. As a result, IVGD suffers risks in overfitting the feature construction task, which usually has a gap with the target downstream task of source localization. Such a gap is neglected by previous methods and leads to suboptimal performance.

To handle this challenge, inspired by a recent work NoisyTune [17] for natural language processing (NLP), we propose a very simple yet effective method named PGD to help better finetune feature construction on the source localization task by adding some noise to the parameters of the feature construction model before finetuning, as Figure 4(b). More specifically, we utilize a matrix-wise perturbing method which adds different uniform noises to different parameter matrices. Unlike NoisyTune, we consider two different ways as follows to add noise.

Denote the parameter matrices in a the feature construction function $f_{\mathbf{W}}$ as $[\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_N]$, where N is the number of parameter matrix types. Denote the perturbed version of the parameter matrix $\mathbf{W}_i$ as $\tilde{\mathbf{W}}_i$, which is computed as follows:

$$\tilde{\mathbf{W}}_i = \mathbf{W}_i + U(0, \alpha) * std(\mathbf{W}_i), \tag{8}$$

where std stands for standard deviation, the function $U(0, \alpha)$ represents uniform distribution noise ranging from 0 to α, and α is a hyperparameter that controls the relative noise intensity. Using this method, parameters in IVGD with higher variance will be added with stronger noise. It is worth noting that this technique is plug-and-play and could be simply implemented by inserting the following PyTorch-style code before finetuning:

```
for name, para in model.named_parameters():
    model.state_dict[name][:] +=
        (torch.rand(para.size()))
        *noise_alpha*torch.std(para)
```

While Equation 8 uses the noise based on the variances of network weights, we also investigate another adding noise method considering the magnitude of network weights:

$$\tilde{\mathbf{W}}_i = \mathbf{W}_i + U(0, \beta) * \mathbf{W}_i, \tag{9}$$

where the function $U(0, \beta)$ represents uniform distribution noise ranging from 0 to β, and β is a hyperparameter that controls the relative noise intensity. Using this method, parameters

in IVGD with larger values will be added with stronger noise. Similarly, it could be implemented by inserting the following PyTorch-style code before finetuning:

```
for name, para in model.named_parameters():
    model.state_dict[name][:] +=
        (torch.rand(para.size()))
        *noise_beta*para
```

IV. Experiments

A. Experimental Settings

1) Datasets: We compare our proposed PGD with the state-of-the-art methods on six real-world datasets in the experiments, of which statistics are shown in Table II and descriptions are outlined as follows:

a. Karate [41]. Karate contains the social ties among the members of a university karate club.
b. Dolphins [41]. Dolphins is a social network of bottlenose dolphins, where edges represent frequent associations between dolphins.
c. Jazz [42]. Jazz is a collaboration network between Jazz musicians. Each edge represents that two musicians have played together in a band.
d. Network Science [43]. Network Science is a coauthorship network of scientists working on network theory and experiment. Each edge represents two scientists who have co-authored a paper.
e. Cora-ML [44]. Cora-ML is a portal network of computer science research papers crawled by machine learning techniques.
f. Power Grid [45]. Power Grid is a topology network of the Western States Power Grid of the United States.

In addition, Figure 5 visualizes the smallest dataset `Karate`.

2) Training Details: For all datasets, we generated diffusion cascades based on the following strategy: 10% nodes were chosen as source nodes randomly, and then the diffusion was repeated 60 times for each source vector. For each cascade, we have a source vector $\boldsymbol{x}$ and a diffusion vector $\boldsymbol{z}_T$. The ratio of the sizes of the training set and the test set was $8:2$.

3) Compared Methods: For perforamance evaluation, four state-of-the-art approaches LPSI [46], NetSleuth [47], GCNSI [15] and IVGD [16] are compared with our proposed, all of which are outlined as follows:

a. LPSI [46]. LPSI is short for Label Propagation based Source Identification. It is inspired by the label propagation algorithm in semi-supervised learning.
b. NetSleuth [47]. The goal of NetSleuth is to employ the Minimum Description Length principle to identify the best set of source nodes and virus propagation ripple [40].
c. GCNSI [15]. GCNSI is a Graph Convolutional Network (GCN) based source identification algorithm. It used the LPSI to augment input, and then applied the GCN for source identification.

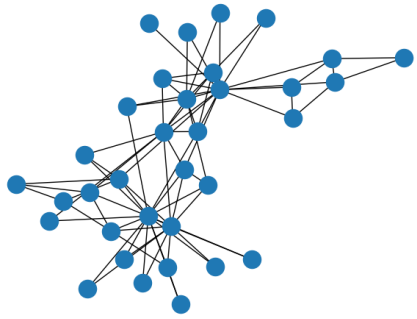


Fig. 5: Visualizations of the real-world dataset `Karate`.

Name	#Nodes	#Edges	Average Degree	Diameter
Karate	34	78	2.294	5
Dolphins	62	159	5.129	8
Jazz	198	2,742	13.848	9
Network Science	1,589	2,742	3.451	17
Cora-ML	2,810	7,981	5.68	17
Power Grid	4,941	6,594	2.669	46

TABLE II: Statistics of six real-world datasets.

d. IVGD [16]. IVGD is an invertible graph diffusion neural network for source localization, which consists of graph residual blocks of feature construction and label propagation to model invertible graph diffusion.

4) Parameter Settings: As we adopt IVGD as our invertible graph diffusion architecture, all the parameter settings follow IVGD [16]. Nonetheless, we additionally perturb the parameters of the neural network before finetuning by adding some noise. The parameters α and β, which are related to the noise intensity, are set to different values according to achieve optimal performance on different datasets. More details would be given and discussed in Section IV-C.

5) Metrics: Five metrics were applied to evaluate the performance:

a. Accuracy (ACC). ACC is the ratio of accurately labeled nodes to all nodes.
b. Precision (PR). PR is the ratio of accurately labeled as source nodes to all nodes labeled as source.
c. Recall (RE). RE defines the ratio of accurately labeled as source nodes to all true source nodes.
d. F-Score (FS). FS is the harmonic mean of Precision and Recall.
e. Receiver operating characteristic curve (AUC). AUC is an important metric to evaluate a classifier given different thresholds.

B. Performance Evaluation

Table III demonstrates the test performance of all compared methods on six datasets, where the best performance is highlighted in bold. Overall, our proposed PGD outperforms all compared methods on all datasets under the optimal hyperparameter settings, which will be further discussed in Section IV-C. Specifically, the ACCs of the proposed PGD on

Dataset	Method	ACC	PR	RE	FS	AUC
Karate	LPSI	0.9559	0.6800	**1.0000**	0.7970	-
	NetSleuth	0.9147	0.5371	0.6833	0.5965	-
	GCNSI	0.7088	0.1150	0.2667	0.1581	-
	IVGD	0.9853	0.8717	**1.0000**	0.9213	0.9975
	PGD (Ours)	**0.9902**	**0.9139**	**1.0000**	**0.9463**	**1.0000**
Dolphins	LPSI	0.9677	0.7790	**1.0000**	0.8717	-
	NetSleuth	0.9306	0.6454	0.7425	0.6904	-
	GCNSI	0.6177	0.1015	0.2548	0.1372	-
	IVGD	0.9935	0.9444	**1.0000**	0.9701	**1.0000**
	PGD (Ours)	**0.9960**	**0.9630**	**1.0000**	**0.9799**	**1.0000**
Jazz	LPSI	0.9035	0.6074	0.9944	0.7371	-
	NetSleuth	0.9222	0.5904	0.6629	0.6245	-
	GCNSI	0.7525	0.0685	0.1280	0.0849	-
	IVGD	0.9980	0.9802	**1.0000**	0.9899	0.9997
	PGD (Ours)	**0.9979**	**0.9803**	**1.0000**	**0.9898**	**0.9998**
Network Science	LPSI	0.9831	0.8525	**1.0000**	0.9202	-
	NetSleuth	0.9595	0.7642	0.8429	0.8016	-
	GCNSI	0.8840	0.0582	0.0135	0.0218	-
	IVGD	0.9946	0.9476	**1.0000**	0.9730	**1.0000**
	PGD (Ours)	**0.9991**	**0.9906**	**1.0000**	**0.9953**	**1.0000**
Cora-ML	LPSI	0.9011	0.5067	0.9993	0.6724	-
	NetSleuth	0.8229	0.1627	0.1793	0.1706	-
	GCNSI	0.8580	0.0970	0.0478	0.0637	-
	IVGD	0.9973	0.9744	**1.0000**	0.9870	**1.0000**
	PGD (Ours)	**0.9963**	**0.9636**	**1.0000**	**0.9815**	**1.0000**
Power Grid	LPSI	0.9673	0.7584	**1.0000**	0.8624	-
	NetSleuth	0.9276	0.6347	0.6986	0.6651	-
	GCNSI	0.7125	0.1022	0.2285	0.1410	-
	IVGD	0.9902	0.9133	**1.0000**	0.9546	**1.0000**
	PGD (Ours)	**0.9949**	**0.9528**	**1.0000**	**0.9758**	**1.0000**

TABLE III: Test performance of all compared methods on six datasets. The best performance is highlighted in bold.

six datasets are all above 0.990, and the PRs are also above 0.91, FSs above 0.94, AUCs above 0.9998. Most importantly, the REs of the proposed PGD constantly achieves 1.

Among the compared methods, the results of LPSI, NetSleuth and GCNSI are not competitive. Specifically, regarding LPSI, its REs achieve 1 almost on all the datasets (except for `Jazz`). However, the ACCs of LPSI on six datasets range from 0.90 to 0.97, PRs range from 0.51 to 0.78, and FSs range from 0.67 to 0.92, which show limited and unstable performance. Similarly, regarding NetSleuth, its ACCs on six datasets range from 0.82 to 0.96, PRs range from 0.16 to 0.76, REs range from 0.18 to 0.84, and FSs range from 0.17 to 0.69, which are even more limited and unstable. Moreover, we observe the bitter failure of NetSleuth at the `Cora-ML` dataset, which indicates that NetSleuth is sensitive to different networks and vulnerable to some specific types. Regarding GCNSI, it shows the worst performance, with ACCs around only 0.70 on all datasets, PRs around only 0.10, REs around only 0.20, and FSs around only 0.15.

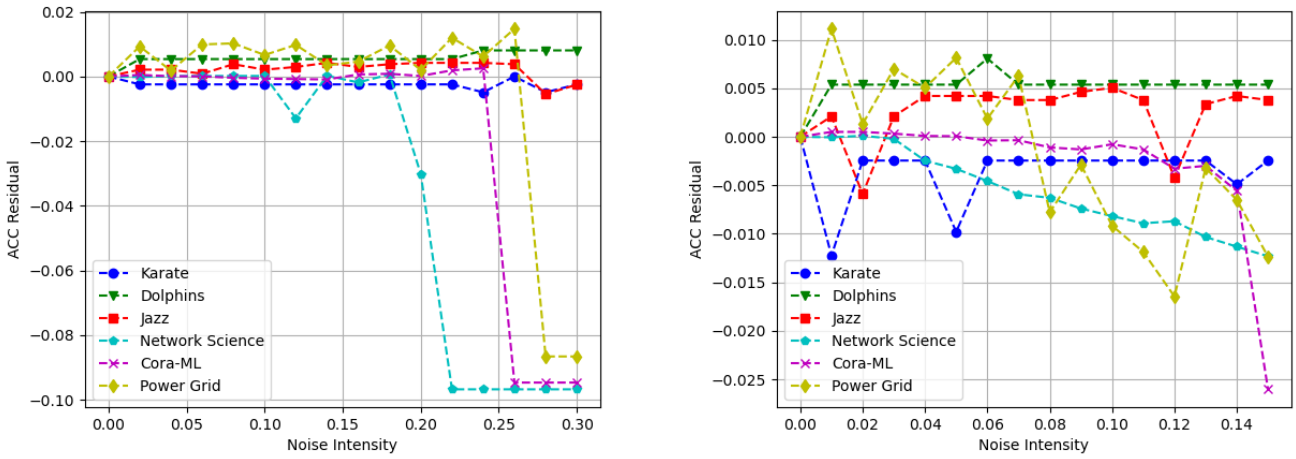
For IVGD, the ACCs on six datasets are above 0.985 (which is 0.015 lower than ours), and the PRs are above 0.87 (which is 0.04 lower than ours), FSs above 0.92 (which is 0.02 lower than ours), AUCs above 0.9975 (which is 0.0025 lower than ours). Therefore, by simply adding some noise to the parameters of the feature construction model before finetuning, our method effectively improves the performance of IVGD.

C. Ablation Study

In this section, we study the influence of the most important hyperparameter in PGD, *i.e.*, α in Equation 8 and β in Equation 9, which control the noise intensity. Figure 6, 7, 8 and 9 show the improvement of ACC, PR, FS, AUC against IVGD, respectively, with regard to different α and β values.

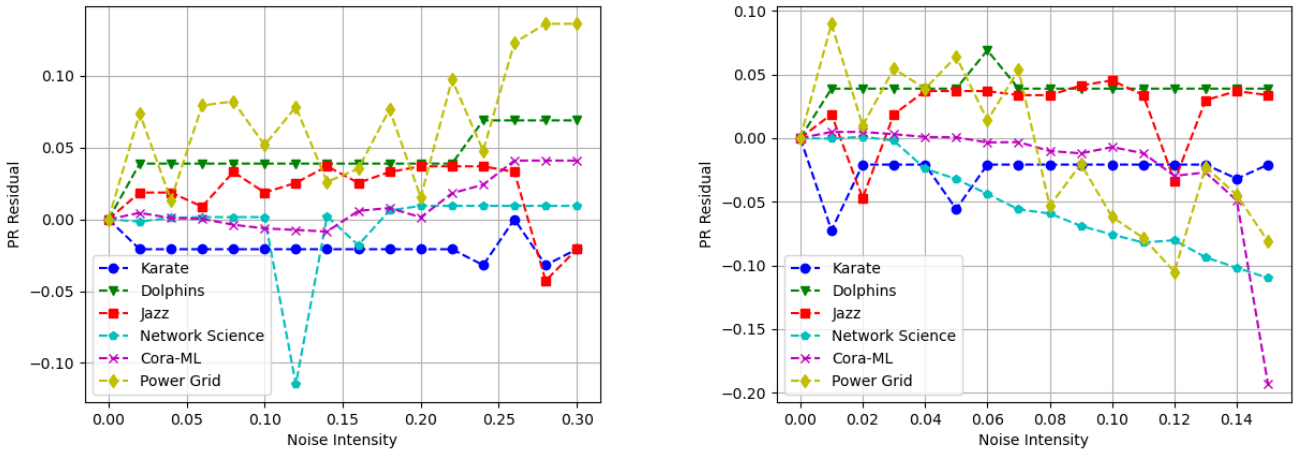
We find that when α or β is too small or too large, the performance is not optimal. This is because when α or β is too small, it is difficult for PGD to do parameter space exploration and overcome the overfitting problem. While when α or β is too large, the useful pretrained knowledge in PGD may be overwhelmed by random noise. For α in Equation 8, values between 0.1 and 0.2 are more suitable for PGD on the six datasets. For β in Equation 9, values between 0.01 and 0.05 are more suitable for PGD on the six datasets.

We additionally find an interesting observation that the two different noise types, which consider either the variance or the magnitude of network weights, lead to different performance. As Figure 8 shows, adding noise based on the magnitude of network weights using β in Equation 9 could better boost the FS performance than that based on the variance using α in Equation 8. On another hand, as Figure 9 shows, adding noise based on the variance could better boost the AUC performance and show stronger robustness than that based on the magnitude. It indicates that adding noise considering both the variance and the magnitude of network weights could be considered and explored in the future.



(a) (b)

Fig. 6: The improvement of ACC against IVGD by our method, with regard to different noise intensities, *i.e.*, (a) α regarding the variance of network weights and (b) β regarding the magnitude of network weights.

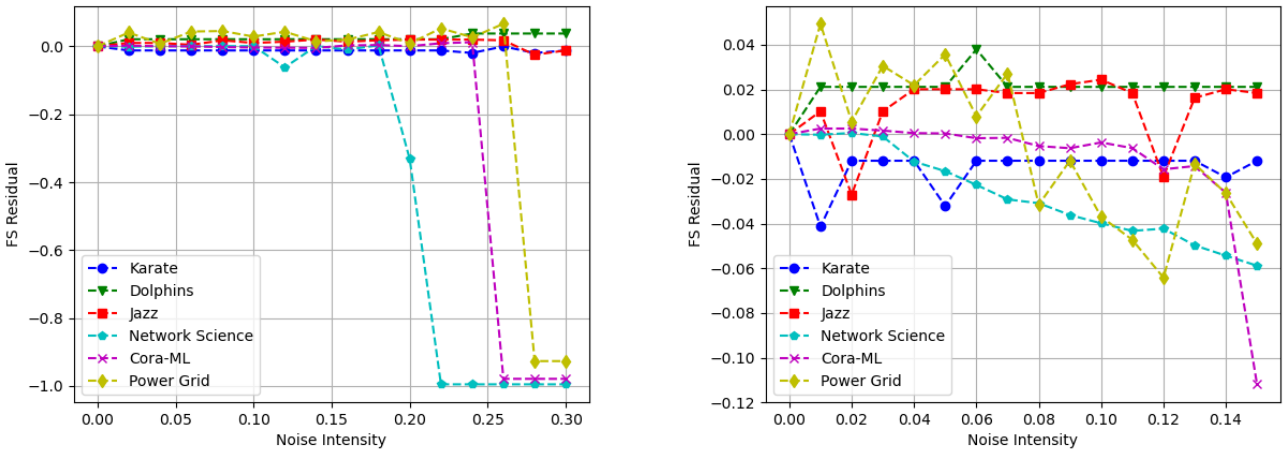


(a) (b)

Fig. 7: The improvement of PR against IVGD by our method, with regard to different noise intensities, *i.e.*, (a) α regarding the variance of network weights and (b) β regarding the magnitude of network weights.

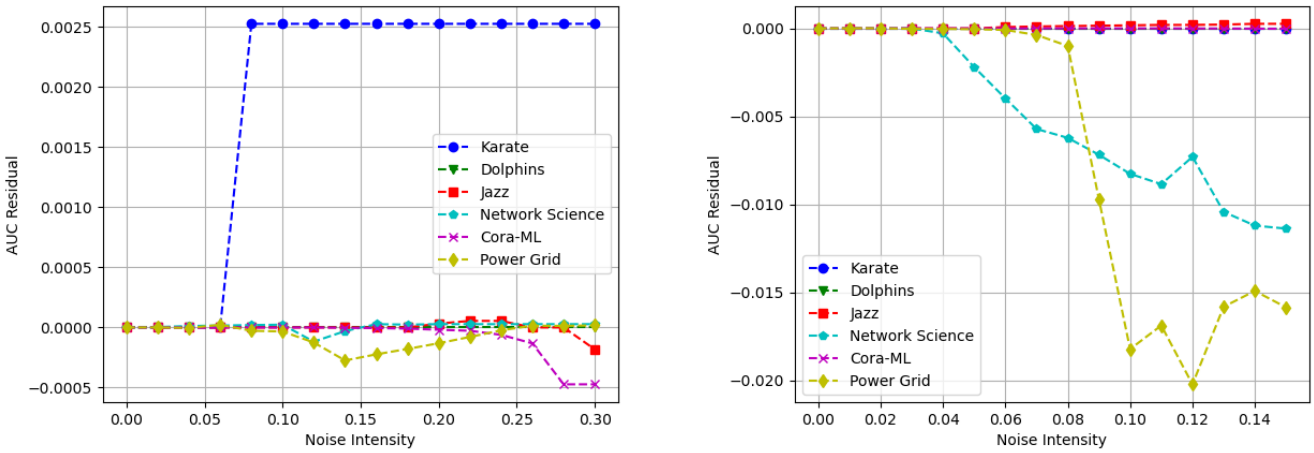
V. Conclusion

In this paper, we focus on source localization, which aims at identifying the source nodes of graph diffusion within a network. To solve the problem that emerging invertible graph diffusion neural network methods have risks in overfitting the feature construction task, we propose a very simple but effective method named PGD, which can help better finetune feature construction on the downstream source localization task by adding a little noise to them before finetuning. In PGD, we propose a matrix-wise perturbing method that adds noise with different intensities to different kinds of parameter matrices in feature construction according to their variances or magnitude. To take a further step from the previous natural language processing work adding noise for finetuning, we design the noise considering the variances and magnitude of network weights, and conduct an ablation study to investigate the performance with different noises. Extensive experiments on six real-world datasets demonstrate the effectiveness of our proposed method and show superior performance against state-of-the-art algorithms. Our ablation study provides a further investigation of the performance improvement regarding different noise types and intensities.



(a) (b)

Fig. 8: The improvement of FS against IVGD by our method, with regard to different noise intensities, *i.e.*, (a) α regarding the variance of network weights and (b) β regarding the magnitude of network weights.



(a) (b)

Fig. 9: The improvement of AUC against IVGD by our method, with regard to different noise intensities, *i.e.*, (a) α regarding the variance of network weights and (b) β regarding the magnitude of network weights.

REFERENCES

[1] Linton Freeman. The development of social network analysis. *A Study in the Sociology of Science*, 1(687):159–167, 2004.

[2] Nicholas JA Harvey, Robert Kleinberg, and April Rasala Lehman. On the capacity of information networks. *IEEE Transactions on Information Theory*, 52(6):2345–2364, 2006.

[3] Björn H Junker and Falk Schreiber. *Analysis of biological networks*. John Wiley & Sons, 2011.

[4] Damon Centola. The spread of behavior in an online social network experiment. *science*, 329(5996):1194–1197, 2010.

[5] Zheng Wang, Chaokun Wang, Jisheng Pei, Xiaojun Ye, and S Yu Philip. Causality based propagation history ranking in social networks. In *IJCAI*, pages 3917–3923, 2016.

[6] Md Saiful Islam, Tonmoy Sarkar, Sazzad Hossain Khan, Abu-Hena Mostofa Kamal, SM Murshid Hasan, Alamgir Kabir, Dalia Yeasmin, Mohammad Ariful Islam, Kamal Ibne Amin Chowdhury, Kazi Selim Anwar, et al. Covid-19–related infodemic and its impact on public health: A global social media analysis. *The American journal of tropical medicine and hygiene*, 103(4):1621, 2020.

[7] Mark EJ Newman, Stephanie Forrest, and Justin Balthrop. Email networks and the spread of computer viruses. *Physical Review E*, 66(3):035101, 2002.

[8] Jae-wook Jang, Jiyoung Woo, Jaesung Yun, and Huy Kang Kim. Mal-netminer: malware classification based on social network analysis of call graph. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 731–734, 2014.

[9] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[10] Lingfei Wu, Peng Cui, Jian Pei, and Liang Zhao. Graph neural networks: Foundations, frontiers, and applications, 2022.

[11] Xiaofeng Gao, Zhenhao Cao, Sha Li, Bin Yao, Guihai Chen, and

Shaojie Tang. Taxonomy and evaluation for microblog popularity prediction. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(2):1–40, 2019.

[12] Fan Zhou, Xovee Xu, Goce Trajcevski, and Kunpeng Zhang. A survey of information cascade analysis: Models, predictions, and recent advances. *ACM Computing Surveys (CSUR)*, 54(2):1–36, 2021.

[13] Mohamed Ahmed, Stella Spagna, Felipe Huici, and Saverio Niccolini. A peek into the future: Predicting the evolution of popularity in user generated content. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 607–616, 2013.

[14] Peng Bao, Hua-Wei Shen, Xiaolong Jin, and Xue-Qi Cheng. Modeling and predicting popularity dynamics of microblogs using self-excited hawkes processes. In *Proceedings of the 24th International Conference on World Wide Web*, pages 9–10, 2015.

[15] Ming Dong, Bolong Zheng, Nguyen Quoc Viet Hung, Han Su, and Guohui Li. Multiple rumor source detection with graph convolutional networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 569–578, 2019.

[16] Junxiang Wang, Junji Jiang, and Liang Zhao. An invertible graph diffusion neural network for source localization. 2022.

[17] Chuhan Wu, Fangzhao Wu, Tao Qi, Yongfeng Huang, and Xing Xie. Noisytune: A little noise can help you finetune pretrained language models better. *arXiv preprint arXiv:2202.12024*, 2022.

[18] Junxiang Wang, Yuyang Gao, Andreas Züfle, Jingyuan Yang, and Liang Zhao. Incomplete label uncertainty estimation for petition victory prediction with dynamic features. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 537–546. IEEE, 2018.

[19] Junxiang Wang and Liang Zhao. Multi-instance domain adaptation for vaccine adverse event detection. In *Proceedings of the 2018 World Wide Web Conference*, pages 97–106, 2018.

[20] Junxiang Wang, Liang Zhao, and Yanfang Ye. Semi-supervised multi-instance interpretable models for flu shot adverse event detection. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 851–860. IEEE, 2018.

[21] Wei Zhang, Wen Wang, Jun Wang, and Hongyuan Zha. User-guided hierarchical attention network for multi-modal social image popularity prediction. In *Proceedings of the 2018 world wide web conference*, pages 1277–1286, 2018.

[22] Guandan Chen, Qingchao Kong, and Wenji Mao. An attention-based neural popularity prediction model for social media events. In *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 161–163. IEEE, 2017.

[23] Wayne Xin Zhao, Hongjian Dou, Yuanpei Zhao, Daxiang Dong, and Ji-Rong Wen. Neural network based popularity prediction by linking online content with knowledge bases. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 16–28. Springer, 2019.

[24] Xueqin Chen, Kunpeng Zhang, Fan Zhou, Goce Trajcevski, Ting Zhong, and Fengli Zhang. Information cascades modeling via deep multi-task learning. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, pages 885–888, 2019.

[25] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1555–1564, 2016.

[26] Qi Cao, Huawei Shen, Keting Cen, Wentao Ouyang, and Xueqi Cheng. Deephawkes: Bridging the gap between prediction and understanding of information cascades. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1149–1158, 2017.

[27] Dongliang Liao, Jin Xu, Gongfu Li, Weijie Huang, Weiqing Liu, and Jing Li. Popularity prediction on online articles with deep fusion of temporal process and content features. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 200–207, 2019.

[28] Shushan He, Hongyuan Zha, and Xiaojing Ye. Network diffusions via neural mean-field dynamics. *Advances in Neural Information Processing Systems*, 33:2171–2183, 2020.

[29] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. Deepinf: Social influence prediction with deep learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2110–2119, 2018.

[30] Jia Wang, Vincent W Zheng, Zemin Liu, and Kevin Chen-Chuan Chang. Topological recurrent neural network for diffusion prediction. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 475–484. IEEE, 2017.

[31] Satoshi Sanjo and Marie Katsurai. Recipe popularity prediction with deep visual-semantic fusion. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2279–2282, 2017.

[32] Cheng Yang, Maosong Sun, Haoran Liu, Shiyi Han, Zhiyuan Liu, and Huanbo Luan. Neural diffusion model for microscopic cascade study. *IEEE Transactions on Knowledge and Data Engineering*, 33(3):1128–1139, 2019.

[33] Lei Ying and Kai Zhu. Diffusion source localization in large networks. *Synthesis Lectures on Communication Networks*, 11(1):1–95, 2018.

[34] Jiaojiao Jiang, Sheng Wen, Shui Yu, Yang Xiang, and Wanlei Zhou. Identifying propagation sources in networks: State-of-the-art and comparative studies. *IEEE Communications Surveys & Tutorials*, 19(1):465–481, 2016.

[35] Devavrat Shah and Tauhid Zaman. Detecting sources of computer viruses in networks: theory and experiment. In *Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 203–214, 2010.

[36] Sushila Shelke and Vahida Attar. Source detection of rumor in social network–a review. *Online Social Networks and Media*, 9:30–42, 2019.

[37] Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. Reversible architectures for arbitrarily deep residual neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[38] Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*, 2018.

[39] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[40] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pages 573–582. PMLR, 2019.

[41] David Lusseau, Karsten Schneider, Oliver J Boisseau, Patti Haase, Elisabeth Slooten, and Steve M Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.

[42] Pablo M Gleiser and Leon Danon. Community structure in jazz. *Advances in complex systems*, 6(04):565–573, 2003.

[43] Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006.

[44] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

[45] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

[46] Zheng Wang, Chaokun Wang, Jisheng Pei, and Xiaojun Ye. Multiple source detection without knowing the underlying propagation model. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), Feb. 2017.

[47] B Aditya Prakash, Jilles Vreeken, and Christos Faloutsos. Spotting culprits in epidemics: How many and which ones? In *2012 IEEE 12th International Conference on Data Mining*, pages 11–20. IEEE, 2012.